MULTI-OBJECTIVE GENETIC PROGRAMMING

FOR DATA VISUALIZATION AND CLASSIFICATION

by

ILKNUR ICKE

A dissertation submitted to the Graduate Faculty in Computer Science in partial

fulfillment of the requirements for the degree of Doctor of Philosophy,

The City University of New York

2011

This manuscript has been read and accepted for the
Graduate Faculty in Computer Science in satisfaction of the
dissertation requirement for the degree of Doctor of Philosophy.

Professor Andrew Rosenberg, PhD

Date — Chair of Examining Committee

Professor Theodore Brown, PhD

Date — Executive Officer

Professor Theodore Brown, PhD

Professor Felisa Vazquez-Abad, PhD

Dr. Steven Gustafson, PhD

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

MULTI-OBJECTIVE GENETIC PROGRAMMING
FOR DATA VISUALIZATION AND CLASSIFICATION

by

Ilknur Icke

Adviser: Dr. Andrew Rosenberg

The process of knowledge discovery lies on a continuum ranging between the human driven (manual exploration) approaches to fully automatic data mining methods. As a hybrid approach, the emerging field of visual analytics aims to facilitate human-machine collaborative decision making by providing automated analysis of data via interactive visualizations. One area of interest in visual analytics is to develop data transformation methods that support visualization and analysis. In this thesis, we develop an evolutionary computing based multi-objective dimensionality reduction method for visual data classification. The algorithm is called Genetic Programming Projection Pursuit (G3P) where genetic programming is utilized in order to automatically create visualizations of higher dimensional labeled datasets which are assessed in terms of discriminative power and interpretability. We consider two forms of interpretability of the visualizations: clearly separated and compact class structures along with easily interpretable data transformation expressions relating the original data attributes to the visualization axes. The G3P algorithm incorporates a number of automated measures of interpretability that were found to be in alignment with human judgement through a user study we conducted.

On a number of data mining problems, we show that G3P generates a large number of data transformations that are better than those generated by a number of dimensionality reduction methods such as the principal components analysis (PCA), multiple discriminants analysis (MDA) and targeted projection pursuit (TPP) in terms of discriminative power and interpretability.

Acknowledgements

The biggest challenge in graduate school is to find an adviser who is both a good researcher and a good mentor. In that regard, my greatest gratitude goes to Dr. Andrew Rosenberg for being a great adviser. Without his help, I would not be sitting here writing this page right now. I thank Dr. Theodore Brown and Dr. Felisa Vazquez-Abad for providing feedback and suggestions and for asking me probing questions. I thank Dr. Steven Gustafson for serving as the outside member of my committee and for his feedback and suggestions especially in the evolutionary computing area.

I am indebted to our Executive Officer, Dr. Theodore Brown and the administrative staff of the Ph.D. program in Computer Science for helping me through the academic maze during my years at the Graduate Center. I also thank Associate Dean Linda Friedman of Zicklin School of Business for granting me the Instructional Technology Fellowship during the final year of my studies.

I would like to thank everyone who participated in the user study I conducted as a part of my research. A number of friends and fellow students deserve special thanks for their support and bringing color to my life: Tanya Poolshup (and her Jack Russell Terrier mix Jack), Veronica Gormley, Deniz Sarioz and Jose Hanchi. I also thank Dr. Jinko Kanno for being a mentor to me via the Internet.

I am grateful to my best friend and roommate Julie Shaw for being there for me through the good times and not so good times.

Finally, I thank my parents Gülten and Kemal İçke and my siblings Sibel and Uğur İçke for their support and love. This thesis is dedicated to them.

Acknowledgement

This research was supported in part by a grant of computer time from the City University of New York's High Performance Computing Research Center under NSF Grants CNS-0855217 and CNS - 0958379.

Dedicated to my family

Contents

List of Figures

List of Tables

List of Algorithms

List of Publications

The author's work in data visualization and evolutionary computing has produced the following publications:

Evolutionary Data Mining and Visual Analytics

1. Icke, I. and Rosenberg A., *Automated Measures for Interpretable Dimensionality Reduction for Visual Classification: A User Study*, VisWeek Posters 2011
2. Icke, I. and Rosenberg A., *Multi-Objective Genetic Programming for Visual Analytics*, In Proceedings of the 14th European Conference on Genetic Programming, EuroGP 2011, volume 6621, pages 323-334, Turin, Italy, 2011
3. Icke, I. and Rosenberg A., *Dimensionality Reduction Using Symbolic Regression*, Genetic and Evolutionary Computation Conference, GECCO 2010, Portland, Oregon, 2010
4. Icke, I. and Rosenberg A. *Multi-Objective Genetic Programming Projection Pursuit for Exploratory Data Modeling*, New York Academy of Sciences, 5th Annual Machine Learning Symposium, October 2010 (poster)

Talks and presentations

1. Icke, I. Multi-Objective Genetic Programming Projection Pursuit for Exploratory Data Modeling, Women in Machine Learning Workshop, December 2010, Vancouver, Canada

Other work in Visual Analytics and Educational Data Mining

1. Icke, I. and Sklar E., *Visual Analytics: A multi-faceted Overview*, Technical report TR-2009005, PhD Program in Computer Science, The Graduate Center, 2009
2. Sklar, E. and Icke, I., *Using Simulation to Evaluate Data-Driven Agents*, Lecture Notes in Artificial Intelligence 5269, Multi-Agent-Based Simulation IX: International Workshop, MABS 2008, Estoril, Portugal, May 12-13, 2008, Revised Selected Papers
3. Icke, I. and Sklar, E., *A Visualization Tool For Student Assessments Data*, From Theory to Practice: Design, Vision and Visualization Workshop at IEEE VisWeek 2008
4. Icke, I. and Sklar, E., *Using Simulation To Evaluate Data-driven Agent-based Learning Partners* , Ninth International Workshop on Multi-agent-based Simulation (MABS'08) at AAMAS 2008
5. Sklar, E., Icke, I., Camacho, C., Liu, W., Salvit J., and Andrewlevich V., *Visualizing Academic Assessment Data*, Workshop of Assessment of Group and Individual Learning Through Intelligent Visualization (AGILeViz) at CSCL 2007

Chapter 1

Introduction

1.1 Motivation

John W. Tukey pioneered the use of statistical graphics for data modeling in 1977 as an alternative to the confirmatory methods which heavily relied on statistical hypothesis testing. He chose to name this new way of analyzing data as 'Exploratory Data Analysis (EDA)' to emphasize that these were methods aimed to help people build hypotheses based on the collected data as opposed to the confirmatory methods which require pre-existing hypotheses to work with. In his seminal work [148], Tukey introduced many of today's well-known statistical graphs such as the scatterplot matrices, box-and-whisker plots and the bubble charts. Tukey's vision of data analysis was impressive given the fact that the computer technology to create the graphics was very primitive compared to what we have available today.[1]

Since the days of being able to manually explore the raw data through visualizations are long gone, the emphasis is on developing automated methods that would explore the space of possible hypotheses and provide the users with the most 'useful' ones with respect to some criteria. The fields 'Visual Data Mining (VDM)' and later 'Visual Analytics' are descendants of EDA, which aim to build a human-machine

[1] Indeed, a video featuring Tukey himself showing his PRIM 9 system in action can be seen at `http://stat-graphics.org/movies/prim9.html`

collaboration for data analysis based on automatically generated visualizations in an interactive environment.

In this thesis, we focus on the analysis of higher dimensional and labeled datasets through automatically generated visualizations. Two or three dimensional scatterplots have been the most common visualization tools for multi-variate data. A scatterplot visualizes the data by projecting it to any two (or three) variables at a time. By inspecting the visualization, the users can draw conclusions about how the variables are related (such as correlated or not). If the data contains multiple groups (class labels), it is also possible to tell if the projection reveals these distinct groups. However, generally it is highly unlikely that projecting on just two (or three) variables would yield any interesting patterns in the data. Many automatic feature extraction techniques have been developed over the years in order to transform the original dataset into a new dataset which reveals a 'useful' pattern. For data visualization, feature extraction techniques replace the original variables with two (or three) variables that are generated with respect to a pre-defined criterion. These methods are more commonly known as the 'dimensionality reduction techniques'.

It is also desired to build predictive models that would learn how to 'classify' the observed data so that when it is given unlabeled data from the same problem domain, it can successfully determine the class labels. Data classification is a large research area in machine learning and data mining fields and many algorithms have been proposed over the years.

Given a data mining problem, two questions arise: 1)'should we try a number of classification algorithms and pick the one with best performance on the dataset?' or 2)'should we pick one classification algorithm and transform the data (extract features) in a way that increases the performance of that classifier?' Generally, either one of these options are selected. In this thesis, we pursue a joint approach that searches for the best feature representation/classifier pairs for a given data mining problem.

1.2 A Multi-Objective Architecture for Visual Classification

The joint process of feature extraction and classifier selection can be performed independently from the number of features to be extracted. Generally, there is no constraint on the number of features to be extracted, provided that classifier accuracy is optimized. The classification problem itself can be seen as feature extraction, where the higher dimensional dataset is transformed into one dimension which identifies the class labels for each data item. Visual classification can be seen as extension of this idea, where the data is transformed into two dimensions which can be visualized as scatterplots and classified using any classification algorithm. Figure 1.1 shows how visual classification works in contrast to the standard data classification scheme.

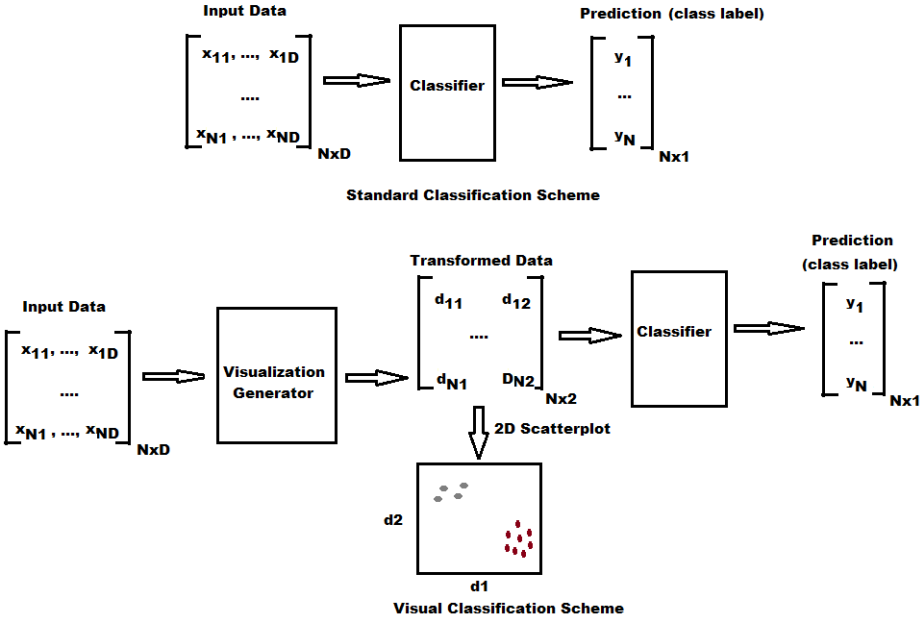


Figure 1.1: Standard classification scheme versus visual classification scheme

The 'visualization generator' component performs dimensionality reduction by transforming the data into two dimensions. Visual classification has the advantage that users can visually inspect the data and understand the class structures within the data. A predictive model can then be developed on the transformed dataset. It is also highly desirable that the users are also presented with the mathematical expressions that transform the data. Interpretability of these expressions are especially important in

understanding how the visualization axes are related to the original features and which variables are important in identifying the distinct class structures on the generated visualization.

However, most dimensionality reduction schemes either do not generate explicit mapping functions or the complexity of the feature transformation functions can not be easily controlled. Moreover, a number of dimensionality reduction techniques do not consider the label information therefore not explicitly aiming to reveal clearly separated class structures.

Considering all these shortcomings associated with the standard practice of data classification, we developed a flexible, multi-objective scheme for visual classification. The algorithm consists of two main components: an evolutionary computing based engine that creates data transformation functions that map the original higher dimensional dataset into 2D visualizations, and a multi-objective assessment scheme that evaluates the quality of these transformations. The evaluation is based on these three objectives:

- Classifiability: Performance of one or more classifiers on the two dimensional representation of the dataset generated using the feature transformation functions. Classifier selection is done by determining the best performing classifier on the transformed data.
- Visual interpretability: The quality of the visualizations from the perspective of how well the class structures are presented in terms of clear separation and compactness of the individual groups.
- Semantic interpretability of the extracted features: The complexity of the feature transformation expressions that map the original dataset into a 2D scatterplot. Complexity of these expressions are important in terms of providing the user with interpretable visualization axes.

We named this algorithm the 'Genetic Programming Projection Pursuit' (G3P) since it utilizes the genetic programming based optimization scheme to search for 'useful' projections of the data in terms of the classifiability, visual interpretability of the view and semantic interpretability of the visualization axes. Figure 1.2 presents a high level abstraction of the G3P architecture.

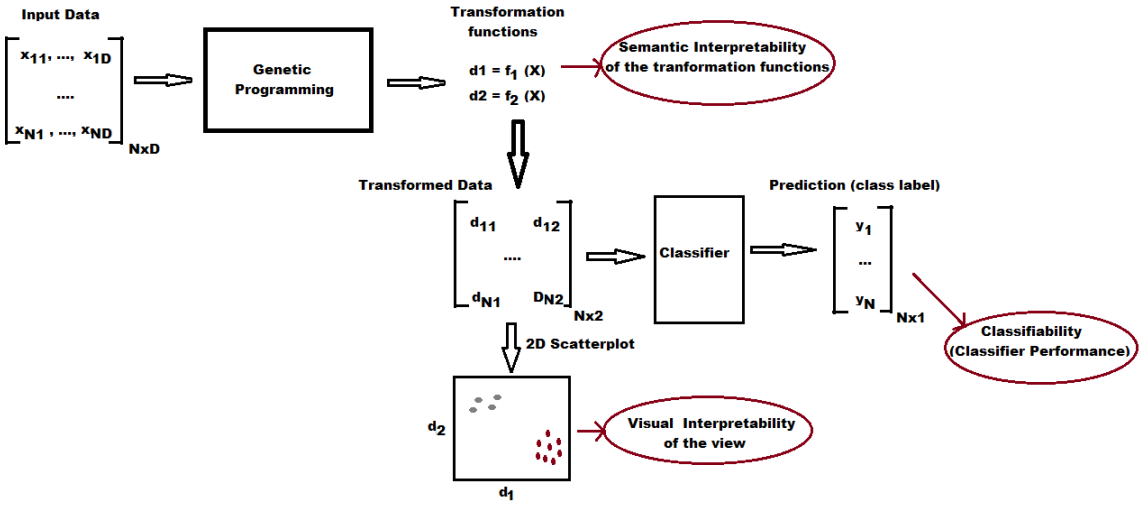


Figure 1.2: Block diagram for the G3P algorithm

1.3 Thesis overview

This thesis is organized as follows:

- Chapter 2 is a review of the following topics: visual analytics and data dimensionality reduction techniques for data visualization, genetic programming and multi-objective optimization schemes.

- Chapter 3 reviews the practice of conducting cross-validation experiments and reporting results in data mining from the perspectives of comparability and reproducibility. Various cross-validation schemes along with statistical significance tests are studied and compared for the general data mining as well as genetic programming based data mining experiments in order to device an experimental protocol.

- Chapter 4 introduces how human interpretability can be incorporated into automatic data mining for model generation. A number of automatic measures for interpretability of data models are presented and compared with human perception of interpretability through user studies.

- Chapter 5 presents a novel multi-objective approach to visual analytics that is called the Genetic Programming Projection Pursuit (G3P) for automatic generation of highly discriminative and interpretable visualizations of higher dimensional labeled datasets for classification tasks. Com-

parisons to a number of standard dimensionality reduction methods are presented and discussed.

- Chapter 6 presents the Multi-Objective Genetic Programming Projection Pursuit (MOG3P) algorithm where a multi-objective optimization scheme is incorporated into G3P. Comparison to G3P is presented and discussed within the context of the general single objective-versus multi objective optimization in evolutionary computing. Incorporation of the best solutions from the standard methods into the MOG3P is introduced as a hybrid method and the improvements are discussed. The effect of over-fitting is investigated and counter-measures are discussed.

- Chapter 7 summarizes the conclusions, discusses the contributions of this work and proposes further research directions.

Chapter 2

Background

2.1 From Exploratory Data Analysis to Visual Analytics

The goal of the whole scientific endeavor is to understand the world around us. In doing this, we rely on our senses to provide us with the observations (data collection) and our brains to make sense out of these observations (knowledge discovery). The invention of computers made it easier to collect large amounts of data and provided various analytical tools for knowledge discovery. The process of knowledge discovery lies on a continuum ranging from manually exploring the data to automatic data mining techniques (figure 2.1).

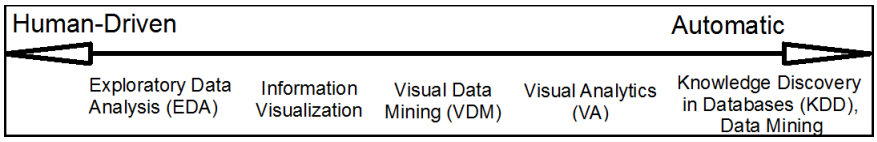


Figure 2.1: The Knowledge Discovery Continuum

Faced with a problem to solve, the first step humans take is to create an abstraction of the problem. In most cases, these abstractions are based on various forms of visualization (such as diagrams and maps). The term Exploratory Data Analysis (EDA) has been introduced by Tukey in 1977 [148]. Tukey suggested the use of statistical graphics as an aid for data analysis. The field Information Visualization emerged in late 1980s and an overwhelming amount of visualization methods have been proposed

since then [159].

On the other side of the data analysis continuum, there have been efforts to employ purely mathematical and statistical methods with little or no emphasis on visualization of the data. Knowledge Discovery in Databases (KDD) has been the popular topic in the 90s, and it is defined as the process of identifying valid, novel, potentially useful, and ultimately understandable structure in data [29]. At the heart of KDD lies a process called Data Mining (DM) which is defined as "*a step in the KDD process concerned with the algorithmic means by which patterns or models (structures) are enumerated from the data under acceptable computational efficiency limitations*" [29]. Focusing heavily on automatically created mathematical models of data comes with a number of challenges. The choice of the right method that matches the characteristics of the data is crucial. The most important challenge is to be able to generate more intuitive and understandable models for the users.

The Visual Data Mining (VDM) concept has been introduced in early 2000s as an interdisciplinary field that aims to utilize human perceptual abilities in data analysis [79]. The idea is that humans might catch the hidden patterns in data which might have been missed by the data mining algorithms provided that they are given interactive tools to visually examine the datasets. The visualization methods employed in VDM borrow techniques from computer graphics and design theory and they are much more complex than statistical graphics used in EDA.

The emerging field of Visual Analytics (VA) "*combines automated analysis techniques with interactive visualizations for an effective understanding, reasoning and decision making on the basis of very large and complex datasets*" [78]. The fundamental aspect of VA is that it proposes a human-machine collaboration for knowledge discovery. Manual exploration of high dimensional and large datasets is not feasible. However, full automation without human intervention is not desirable either. Therefore, the challenge is to determine how much of the analysis can be done automatically and what kinds of decisions can be left for the users to make. One major reason why users do not prefer totally automated systems is because they would like to intervene and make sure that the system is generating interpretable models besides being accurate. An accurate model is useless unless the user can understand

how it works. If automated 'measures of interpretability' could be devised and incorporated into the system, it would be possible to relieve the users from the burden of supervising the model generation process closely. This is the main goal of this thesis, in which we present an evolutionary computing method that generates easily interpretable visual models for data classification based on a number of automated interpretability measures.

The rest of this section is organized as follows: the relationships between data visualization and mathematical models for data transformations are discussed in section 2.2, the concept of interpretability of scatterplots is discussed in section 2.3, an overview of the evolutionary computing method that is known as genetic programming is given in section 2.4. The final section presents some background information on the datasets that are used in the experiments throughout this thesis.

2.2 Relationships Between Data Visualization, Feature Extraction and Dimensionality Reduction

In this section we present a number of ways for visualization of multivariate data and then discuss various mathematical models that transform a given dataset in a way that 'useful' patterns in the data are revealed.

2.2.1 Overview of Visualization Methods for Multivariate Data

In this section, we briefly introduce a number of visualization methods that are commonly in use for visualizing multivariate and possibly labeled datasets. Scatterplot is the most common visualization type which can reveal the relationships between the data axes, the groupings within the data or both. Since the cartesian coordinate system (orthogonal axes) is used in visualization, it is possible to display only two or three variables on a single scatterplot. Multiple scatterplots are used in order to display all of the variable pairings for small datasets. Figure 2.2 presents the scatterplot matrix for a small dataset (the UCI Iris dataset [3]) that contains four measurements characterizing the three species of a flower.

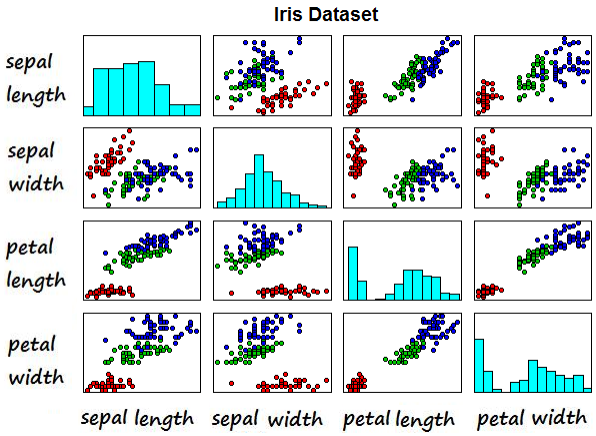


Figure 2.2: Scatterplot matrix for the Iris dataset

Inselberg introduced a technique known as *parallel coordinates* in order to overcome the limitation of being able to visualize two or three variables at a time (figure 2.3). In this technique, coordinate axes are drawn in parallel instead of orthogonally [72, 71]. The ordering of the dimensions might affect the usefulness of the visualization. Ankerst et al. propose a technique that clusters the data dimensions based on their similarities to enhance visualizations in [10].

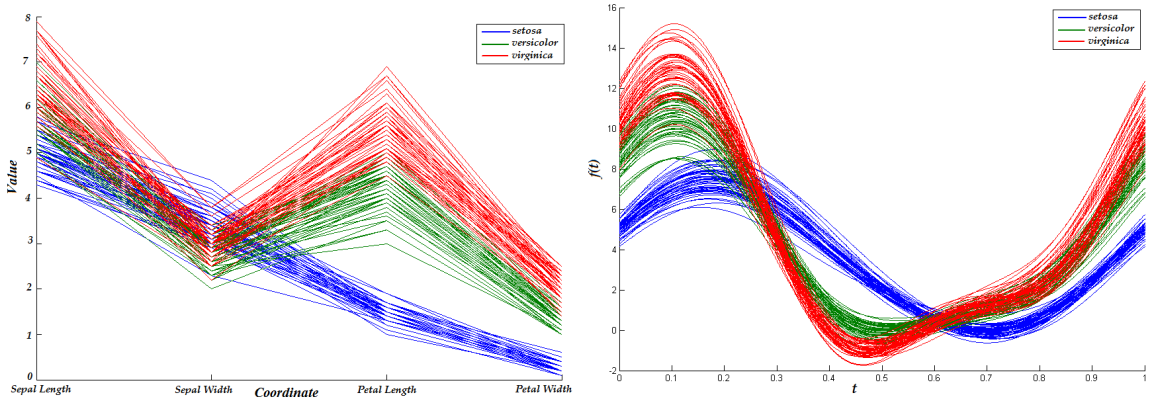


Figure 2.3: Parallel coordinates (left) and Andrews curves (right) visualizations for the Iris dataset

Andrew's Plots method [9] projects each data item $X = (x_1, x_2, ..., x_N)$ from vector space into trigonometric function space. The variables (x_i) of each observation become the coefficients of the following

Fourier series:

$$f(t) = \frac{x_1}{\sqrt{2}} + x_2 sin(\pi t) + x_3 cos(\pi t) + x_4 sin(2\pi t) +$$
$$x_5 cos(2\pi t) + x_6 sin(3\pi t) + x_7 cos(3\pi t) + ...$$

where $0 \leq t \leq 1$. As it can be seen from the equation, the ordering of the variables affect the shape of the curve.

Introduced by Herman Chernoff in 1973, Chernoff faces [35] is by far the most famous data picturing method. It is possible to project (map) up to 18 data features onto various face features (such as size and curvature of the face, position of mouth, eyes, nose, size of features). Faces are special visual items because humans are naturally wired to recognize the faces, but the underlying mechanisms are still not well understood. Therefore, it remains a challenge to assign the data features onto the *appropriate* face features in order to maximize the effectiveness of this method.

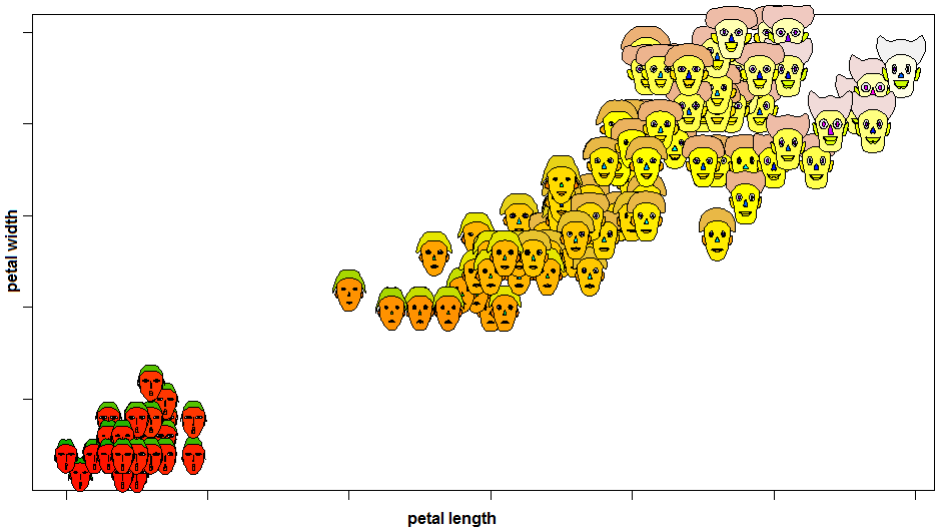
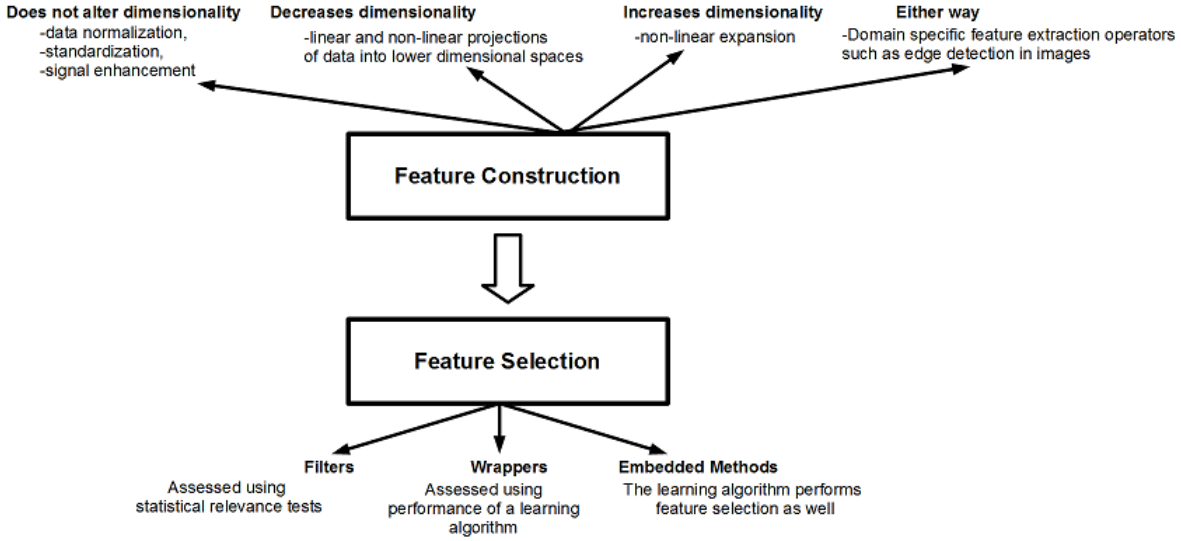


Figure 2.4: Chernoff Faces visualization for the Iris dataset

2.2.2 Feature Extraction

Feature extraction is *the process of finding a good representation for a given dataset so that the outcome of the machine learning process is optimized*. Guyon and Elisseeff give an overview of feature extraction in [64]. They define feature extraction as a two-step process: *feature construction* and *feature selection*.



Feature construction is further divided into two classes: integrated into the model building process as in hidden units of neural networks computing internal representations of variables, and as preprocessing step before model building. The authors identify four kinds of feature construction techniques with respect to how they change data dimensionality: 1) *Does not alter dimensionality:* data normalization, standardization, signal enhancement, 2)*Decreases dimensionality:* linear and non-linear space embedding methods for projecting data into lower dimensional spaces, 3) *Increases dimensionality:* discretization, non-linear expansion, 4)*Either way:* Domain specific feature extraction operators such as edge detection in images

For feature selection step, the authors describe three aspects: 1) search technique, 2) evaluation criterion, and 3) assessment method. Based on these aspects, feature selection methods can be classified into three approaches which are called filters, wrappers and embedded methods respectively (Table 2.1).

Approach	Search method	Evaluation criterion	Assessment method
Filters	Exhaustive search Heuristic/Stochastic Search Single feature ranking nested subset, forward selection/backward elimination	Statistical Tests	Single feature relevance Relevance in context Feature subset relevance
Wrappers	Exhaustive search Heuristic/Stochastic Search Single feature ranking nested subset, forward selection/backward elimination	Cross validation Performance bounds	performance of a learning machine
Embedded Methods	Single feature ranking nested subset, forward selection/backward elimination	Cross validation Performance bounds	performance of a learning machine

Table 2.1: Feature selection approaches from [64]

Filter methods do not utilize a classifier to assess the usefulness of the features, instead they use various statistical measures of the relationships between the features and the class variable. Wrapper methods use classifier accuracy as the criterion to assess the utility of the features. In embedded approaches, the classifier explicitly contains mechanisms to perform feature selection (such as decision trees) and feature construction (such as multilayer perceptron neural networks).

2.2.3 Dimensionality Reduction

In this section, we present an overview of various techniques from the machine learning literature that are used for dimensionality reduction. Table 2.2 presents a summary of these methods with respect to four aspects:

1. The nature of transformation identifies between the linear and nonlinear methods. Linear methods are considered simpler and easier to compute using matrix operations. However, they might not as accurate as the nonlinear methods.

2. Consideration of the label information discriminates between the supervised (labels are used) and unsupervised (labels are ignored) methods.

3. Geometry preservation is an explicit objective for a number of methods which aim to transform the data in a way that preserves the distances between the data items. Global methods try to preserve the whole structure while the local methods are more flexible.

4. Availability of an explicit mapping function can be useful in examining how the transformed features are related to the original data attributes.

Method	Nature of Transformation		Label Information		Geometry Preservation		Explicit Mapping	
	Linear	Nonlinear	Supervised	Unsupervised	Global	Local	Yes	No
Principal Components Analysis (PCA)	√			√	√		√	
Kernel PCA		√		√	√		√	
Multiple Discriminants Analysis (MDA)	√		√		√		√	
Kernel MDA		√	√		√		√	
Multidimensional Scaling (MDS)		√		√	√			√
Exploratory Projection Pursuit (EPP)	√		√	√	√		√	
Targeted Projection Pursuit (TPP)	√		√		√		√	
ISOMAP		√		√	√			√
Locally Linear Embedding (LLE)		√		√		√		√
Laplacian Eigenmaps		√		√		√		√
Neural Network Based MDS (SAMANN)		√		√	√		√	
Neural Network Based MDA (NDA)		√	√		√		√	
Neighbor Retrieval Visualizer (NeRV)		√		√		√		√
Supervised NeRV (SNeRV)		√	√			√		√

Table 2.2: Summary of dimensionality reduction methods

Linear Methods Linear dimensionality reduction techniques utilize linear algebra to find a lower dimensional representation of the data by projecting the dataset based on various criteria. Principal Components Analysis (PCA) is the earliest and still the mostly used technique and it was invented by Karl Pearson in 1901. PCA finds a lower dimensional representation such that maximum variation in the data is still retained [75]. Given a dataset $D = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \in R^d$, PCA algorithm tries to find:

$$\underset{a \in R^d, ||a||=1}{\operatorname{argmax}} \ (aSa'); \tag{2.1}$$

where $S = \frac{1}{n-1}(\mathbf{x}_i - \bar{\mathbf{x}})'(\mathbf{x}_i - \bar{\mathbf{x}})$ is the sample covariance matrix. From linear algebra, a is the eigenvector corresponding to the largest eigenvalue λ_1 of matrix S. This is the direction of maximum variance in the dataset. The second direction that captures the most variance is the vector orthogonal to a that corresponds to the second largest eigenvalue. The magnitude of each eigenvalue indicates the percentage of the total variance it represents. The full set of eigenvalue,eigenvector pairs can be found using singular value decomposition of S. Dimensionality reduction is achieved by projecting the data onto first k principal components where (k $<<$d).

PCA is a computationally expensive algorithm for large datasets. The random projections (RP) algorithm has been introduced as a computationally efficient dimensionality reduction method that gives sufficiently accurate results. RP is based on the Johnson-Lindenstrauss lemma which states that *if points in a vector space are projected onto a randomly selected subspace of suitably high dimension, then the distances between the points are approximately preserved* [22]. RP reduces the dimensionality

of a d-dimensional dataset into a k-dimensional dataset (k << d) by:

$$D'_{k\times N} = R_{k\times d} D_{d\times N}$$

where a common choice to build the random projection matrix R is a simple distribution suggested by Achlioptas [5]: $r_{ij} = \sqrt{3} \cdot \{+1$ with probability $\frac{1}{6}, 0$ with probability $\frac{2}{3}, -1$ with probability $\frac{1}{6}\}$

The Exploratory Projection Pursuit (EPP) algorithm has been proposed by Friedman and Tukey [58] and it is a general linear dimensionality reduction framework that is iterative and looks for *interesting* projections. The degree of interestingness is quantified by a projection index. For example, PCA algorithm can be implemented as projection pursuit if the projection index is as in equation 2.1. Linear projections based on Fisher's linear discriminant analysis (LDA) have been proposed in [18, 91] for classification tasks and various indices for clustering tasks have been introduced in [25].

The Multiple Discriminants Analysis (MDA) (also known as the Canonical Variates Analysis) is a supervised method that takes the label information into account [47]. For a c-category classification problem, MDA finds a $c-1$ dimensional representation of the data such that the separation between the categories is maximized. This is achieved by maximizing the ratio of between-class scatter matrix to within-class scatter matrix. Similar to PCA, each new feature is a linear combination of the original features.

The Targeted Projection Pursuit (TPP) algorithm introduced by Faith et al. [54] is also a supervised method that aims to project the data in a way that different groups are maximally separated from each other. Instead of optimizing a measure of class separation, the authors determine a target view and then find a linear projection that maps the original data to the target view as closely as possible. For a c-category dataset, the target view is defined as a c-simplex in the c-dimensional space where each class is equally separated from each other. The authors train a single layer feedforward neural network to determine the weights of the projection that maps the input data into the target view as closely as possible. If $c > 2$, in order to generate the final visualization from c-dimensions to 2D, the algorithm uses PCA or a search based method to reduce the dimensionality of the target view.

Nonlinear Methods Nonlinear versions of the PCA and LDA algorithms utilizing the kernel trick have also been proposed [160]. These methods use kernel functions to map the data into higher dimensional spaces where the principle components (PCA) or the separation boundaries (LDA) are linear as opposed to being nonlinear in the original space. Multidimensional scaling (MDS) algorithm searches for a mapping of the input data onto a k-dimensional space that preserves the inter-point distances. Given a matrix of inter-point distances between the original data points, where $d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||$, a variant of this algorithm named Sammon's mapping minimizes the following:

$$\sum_{i \neq j} \frac{(d_{ij} - ||\mathbf{z}_i - \mathbf{z}_j||)^2}{d_{ij}}$$

where $\mathbf{z}_i, \mathbf{z}_j$ are the new feature vectors in k-dimensional space.

A common intuition shared by a number of researchers in machine learning community that most high dimensional datasets have an inherent lower dimensional representation that may not be a linear subspace. This inherent structure is called a manifold since it is modeled as a low dimensional shape that is embedded into the high dimensional feature space by folding (therefore, non-linear). This research area is known as manifold learning and a large number of algorithms have been reported over the last decade that aim to unfold the manifolds into their inherent (intrinsic) dimensions. The ISOMAP algorithm [145] proposed by Tenenbaum *et* al. is one of the earliest algorithms for manifold learning. The algorithm first computes a k-Nearest Neighbor Graph for the dataset (size N) where the edges are assigned weights as the Euclidean distances between the points and then compute the all-points shortest path distances matrix M. Then, given distance matrix M, it performs MDS to map the data into 2D Euclidean space (figure 2.5).

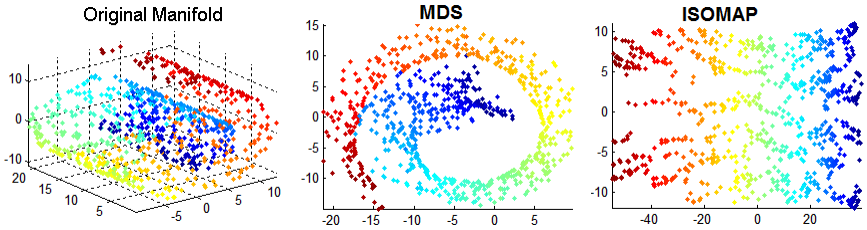


Figure 2.5: MDS and the ISOMAP methods on the Swiss Roll data

Overviews of various manifold learning algorithms can be found in [33, 30]. The disadvantage of

these algorithms is that, they do not generate explicit functions defining the new dimensions with respect to the original data features, therefore it is generally difficult to interpret the reduced dimensions.

Neural network implementations of discriminants analysis (NDA Network) and Sammon's mapping (SAMANN Network) have been proposed in [102]. Though these algorithms suffer from all the issues related to training a neural network, they unify data projection and different optimization criteria under one framework. Moreover, after training is done, transformation functions that would map the original data with respect to the optimization criterion can be extracted from the neural network which is a great advantage over many of the nonlinear methods presented here. The Neighbor Retrieval Visualizer (NeRV) and Supervised Neighbor Retrieval Visualizer (SNeRV) methods [154] take an information retrieval approach to data visualization. Given a higher dimensional dataset, the NeRV finds a 2D view where the neighborhoods of data points are preserved in terms of precision and recall measures. The SNeRV is an extension of NeRV which also considers the class labels.

Another method to perform nonlinear dimensionality reduction is constructive induction via evolutionary computing methods where new features are created from the original features using nonlinear functions. The advantage of the evolutionary methods is that one can introduce constraints on the form and complexity of the feature transformation functions generated by the algorithm. Moreover, instead of creating one model at a time, a population of models are created and evaluated at once. A general overview of evolutionary algorithms in data mining can be found in [57]. Since the focus of this work is genetic programming, we give an overview of genetic programming methods in dimensionality reduction in a separate section (section 5.1).

2.3 Interpretability of Visualizations

In this section, we focus on the interpretability of scatterplots that are generated by transformation of higher dimensional labeled data into a two dimensional representation. We study the interpretability from two perspectives: the interpretability of the view and the interpretability of the visualization axes.

2.3.1 Assessing Quality of Visualizations (Scatterplots)

In an exploratory data analysis setting, the task of selecting interesting [58] or good views of data becomes more challenging as the dimensionality increases. For sufficiently high dimensional datasets, manual exploration of the space of views is impractical. In the case of labeled data, the degree of interestingness is related to how easy it is to tell the classes apart from each other by inspecting the visualization.

Lee at al. present a measure for exploratory projection pursuit of labeled data that is based on Fisher's Linear Discriminant Analysis method in [91]. The VizRank algorithm proposed by Leban et al. searches for *informational* 2D projections of datasets that are evaluated by a k-Nearest Neighbor classifier [90]. The authors claim *almost perfect* agreement between the human judgement and the VizRank algorithm through a user study conducted using six datasets. Sips et al. propose two measures based on the notion of *class consistency* in [137]. One measure is based on preservation of closeness to class centroids after projection, and another is based on the entropies of the spatial distributions of the classes. The authors report a user study on a number of datasets with varying number of classes. They claim that their proposed measures are in alignment with human judgement in terms of finding all views that were labeled as good views by the participants. Tatu et al. propose two measures to evaluate the degree of separation on scatterplots of labeled data in [143]. Tatu et al. report a user study in [144] that compares four visual quality measures that have been proposed in [137] and [143]. The authors suggest that a combination of measures might be worth investigating.

2.3.1.1 Wrapper Methods: Using Classifiers

Since the goal of a classifier is to tell the classes apart, high accuracy on the generated 2D data would also mean a good view of the data. Therefore, classifier accuracy can be a quality measure to assess scatterplots. However, classification algorithms have different underlying principles. Therefore, each algorithm displays a different decision boundary characteristic that is related to how the algorithm works. Figure 2.6 shows the decision boundaries of various classifiers on an example 2D point cloud.

These images were generated using Weka [4] and each classification algorithm was trained on this example dataset with the default parameter settings.

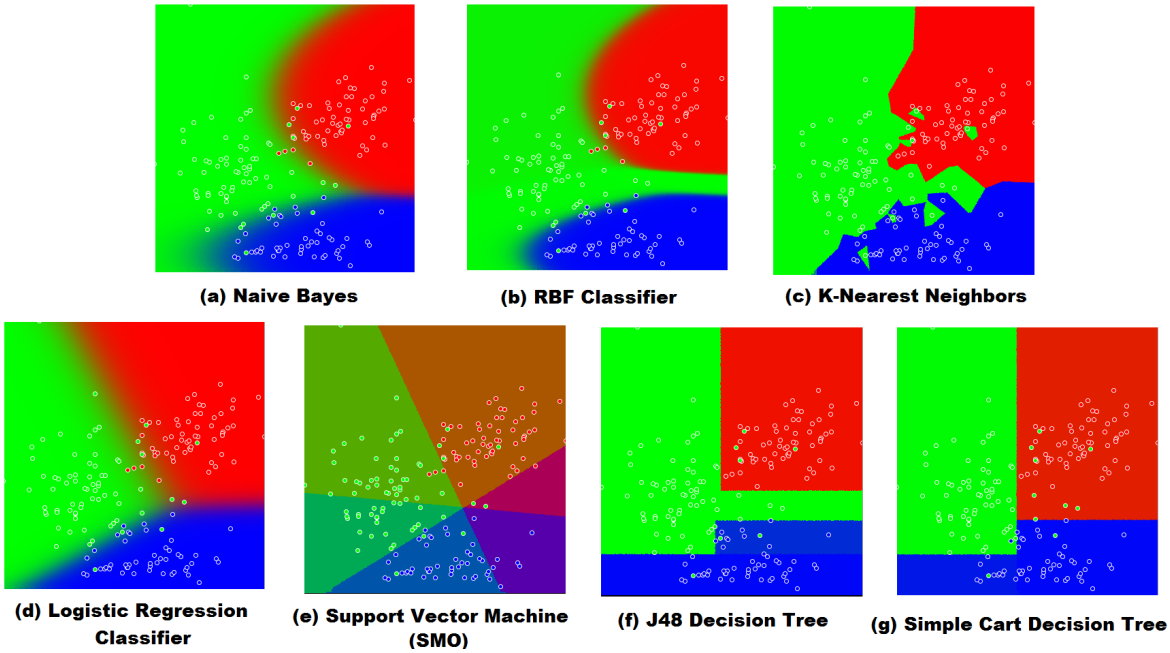


Figure 2.6: Classifier boundaries on a 2D view of labeled data containing 3 groups. Members of each group are displayed in separate colors.

Characteristics of each algorithm can be contrasted to how humans would draw the boundaries between the classes on the views. Some algorithms have linear boundary characteristics. The boundary generated by a support vector machine depends on the kernel it uses, it can be linear or nonlinear. The boundaries by the k-nearest neighbors are generally irregular especially for small k.

2.3.1.2 Cluster Validity Indices

Data clustering is a well-known machine learning problem of categorizing multi-dimensional data into natural groupings such that items that are in the same group are more similar to each other than items from other groups. A number of methods have been proposed in order to quantify the quality of the outcome of clustering algorithms. In the case of 2D data, cluster validity indices can be used as measures of interpretability of the visualizations. In this section, we discuss three measures that were used in our experiments. The unifying theme of these cluster validity indices is that they all aim to

measure compactness and well-separation of the class structures using a distance measure and they are susceptible to outliers. Detailed overview of validity indices can be found in [65, 122].

C Index (I_C) The C Index is a cluster validation index defined in [68]:

$$I_C = \frac{SD - SD_{min}}{SD_{max} - SD_{min}}$$

where SD is the total sum, for all classes, of pairwise distances between samples of the same class (total p distances), SD_{min} and SD_{max} are the sums of p smallest/largest pairwise distances across the whole dataset. The C index returns values between 0 and 1. Smaller values of I_C indicate more compact and better separated class structures.

Davies-Bouldin Index (I_{DB}) The Davies-Bouldin Index is a measure of compactness and well separation of clusters and it was proposed in [39]:

$$I_{DB} = \frac{1}{n}\sum_{i=1}^{n} min_{i \neq j}\{\frac{\delta(X_i, X_j)}{\Delta(X_i) + \Delta(X_j)}\}$$

where $\Delta(X_i)$ is intra-class distance for class i and $\delta(X_i, X_j)$ is inter-class distance for classes i and j. Smaller values of I_{DB} indicate more compact and better separated cluster structures. In our experiments, we normalized the I_{DB} values to [0,1] range.

Dunn's Index (I_{Dunn}) The Dunn's index is a measure of compactness and well separation of clusters and it was proposed in [48]:

$$I_{Dunn} = min_{1 \leq i \leq n}\{min_{1 \leq j \leq n, j \neq i}\{\frac{\delta(X_i, X_j)}{max_{1 \leq k \leq n}\{\Delta(X_k)\}}\}\}$$

where δ, Δ are defined as above. Smaller values of I_{Dunn} indicate more compact and better separated class structures. In our experiments, we normalized the I_{Dunn} to [0-1] range.

2.3.1.3 Other Measures

In this section, we discuss three methods that were introduced in visual analytics literature. The Class Consistency Measure and the 2D Histogram Density Measure have been reported to be the closest matches to human perception amongst the four proposed visual quality measures through a user study [144].

LDA Index (I_{LDA}) The LDA index is based on Fisher's discriminant analysis and has been introduced in [91] for exploratory projection pursuit for classification problems:

$$I_{LDA} = \frac{|W|}{|W+B|}$$
$$B = \sum_{c=1}^{k} n_c(\bar{X}_c - \bar{X})(\bar{X}_c - \bar{X})'$$
$$W = \sum_{c=1}^{k}\sum_{j=1}^{n_c}(X_{cj} - \bar{X}_c)(X_{cj} - \bar{X}_c)'$$

where X_{cj}: data points, $\bar{X}_c$ and $\bar{X}$: group and dataset centroids, k: number of groups (classes), n_c: number of points in group c, B: between-group sum of squares, W: within-group sum of squares Smaller values of I_{LDA} indicate more compact and better separated class structures.

Class Consistency Measure (CCM) The Class Consistency Measure (CCM) has been proposed in [137] and is based on the preservation of closeness to class centroid after a projection from the original data space into a 2D view. A data point is said to be *inconsistent*, if the projection places it closer to a class centroid other than its own. CCM scores each view with respect to how many consistent points it contains:

$$CCM = 1 - \frac{\sum_{c=1}^{k} CC(x_c)}{M}$$
$$CC = \begin{cases} 1, \text{ if } d(x_c, \bar{x}_c) < d(x_c, \bar{x}_i), 1 \leq i \leq k, i \neq c \\ 0, \text{ otherwise} \end{cases}$$

where M is the total number of data points, k is the number of classes, x_c is a point in class c and $\bar{x}_c, \bar{x}_i$ are projections of the centroid of classes c and i respectively. The CCM returns values between 0 and 1 where smaller values indicate more consistent views.

2D Histogram Density Measure (2D-HDM) The Histogram Density Measure (HDM) which has been proposed in [143] is a measure of class separation based on 2D histograms computed on 2D scatterplots of data. A weighted sum of entropy of each bin and its immediate neighbors (u_c) is computed as follows:

$$2D - HDM = \frac{1}{Z} \sum_{x,y} \sum_{c} u_c (- \sum_{c} \frac{u_c}{\sum_c u_c} log_2 \frac{u_c}{\sum_c u_c})$$

$$\frac{1}{Z} = \frac{1}{log_2 M \sum_{x,y} \sum_c u_c}$$

where $\frac{1}{Z}$ is a normalization factor in order to confine the values within the [0-1] range and smaller values indicate better data separation. The choice of bin size influences how this measure scores the views.

2.3.2 Interpretability of Visualization Axes

In most dimensionality reduction and feature construction methods, the interpretability of visualization axes or the feature transformation functions are not considered. In the case of projection pursuit, the projection functions are given as weighted linear combinations of the original features. Morton defines the interpretability of these projection functions in terms of parsimony (simplicity) and proposes rotation and entropy based methods to simplify the coefficients of the linear projections while preserving the *interesting* view [105]. El-Arini et al. present a dimensionality reduction method that searches over scatterplots generated by simple arithmetic expressions of the original features and assessed by accuracy of a Bayesian classifier [80]. The authors claim that expressions containing more than one or two features become less interpretable. However, no empirical study has been reported to justify this intuition.

2.4 Genetic Programming

Genetic programming (GP) is one of the many 'nature inspired' optimization techniques such as the ant colony optimization, particle swarm optimization and simulated annealing (see for example [55, 34, 36] for various examples of this computing metaphor). The first GP related research surfaced in the 80s. Smith's PhD thesis [141] on genetic adaptive algorithms is considered to be the first GP related work. Papers published by Cramer in 1985 [38] and later in 1987 by Schmidhuber [130] are highly cited as the earliest GP work utilizing variable length string based and tree based representations. However, John Koza is considered to be the predominant figure behind GP. In 1992, Koza published his seminal book on GP as way of automatically 'evolving' computer programs as candidate solutions for various problems [83]. GP is a member of the population based stochastic algorithms that are known as the 'evolutionary algorithms' along with the genetic algorithms, evolutionary programming and evolution strategies [50].

There are many optimization frameworks such as the exhaustive search which searches the space in an ordered manner, random search which explores the space randomly and the gradient based methods (hill climbing) which require a differentiable function to optimize. The gradient based methods easily get stuck at sub-optimal solutions (local optima), however, stochastic variants have been introduced to remedy the shortcomings of the original algorithm. Evolutionary algorithms were inspired by the natural evolutionary process which promotes the "survival of the fittest" in a population and creating new individuals through breeding. They do not require a differentiable function to optimize. They sample the space randomly but more efficiently than a pure random search thanks to the genetic operators. The crossover operator works like hill climbing; it creates offspring individuals from the existing fit individuals by creating hybrid individuals which are similar to their parents (exploitation). The mutation operator is more like random search; it can dramatically change an individual and shift the search towards different areas of the search space (exploration). Since there is a population of solutions rather than just one solution, multiple areas of the search space are explored in parallel which makes it less likely for the

algorithm to get stuck at the local optima. The algorithm is stochastic since the genetic operators act on the population with respect to predefined probabilities. In summary, the evolutionary methods offer more potential for non-differentiable search spaces with many optimal solutions compared to the classical methods such as hill climbing. Detailed discussions and comparisons between the evolutionary and the standard optimization methods can be found in the references such as [34].

In this section, we present an overview of genetic programming starting from what it does and how it works and then briefly overview the possible theoretical explanations on why it works and discuss a number of shortcomings and remedies proposed in the GP literature. Finally, we discuss the multi-objective framework within the context of evolutionary computing which allows optimization using multiple and often conflicting objectives.

2.4.1 The Machinery of Genetic Programming

Koza adopted an 'expression tree' representation for the computer programs that the GP builds. These expression trees can be as simple as mathematical expressions consisting of arithmetic operators and operands or full-fledged programs solving complex problems where the nodes of the trees can be functional symbols corresponding to any conceivable task. Like in every optimization problem, GP also considers a criterion to optimize which is embedded into the algorithm through a fitness function.

Two common GP examples are the Symbolic Regression and the Santa Fe Trail problems. Symbolic Regression is a standard statistics/machine learning problem where GP is used to generate mathematical expressions that fit a given set of data points. The Santa Fe Trail problem is a reinforcement learning problem where the actions of an agent is motivated by maximizing the reward it receives from the environment based on its actions.

Figure 2.7 shows an example of Symbolic Regression. A 3D dataset generated using $f_o = {x_0}^2 * x_1$ with added Gaussian noise is shown on the left. The expression tree (f_e: estimated function) generated by the best GP solution is given on the right. The fitness function is the root mean squared error (RMS) which computes how close the estimated values are to the original values. Symbolic Regression is

a major topic in GP community with real life applications such as scientific data modeling [131], drug discovery [153] and chemical process modeling [157].

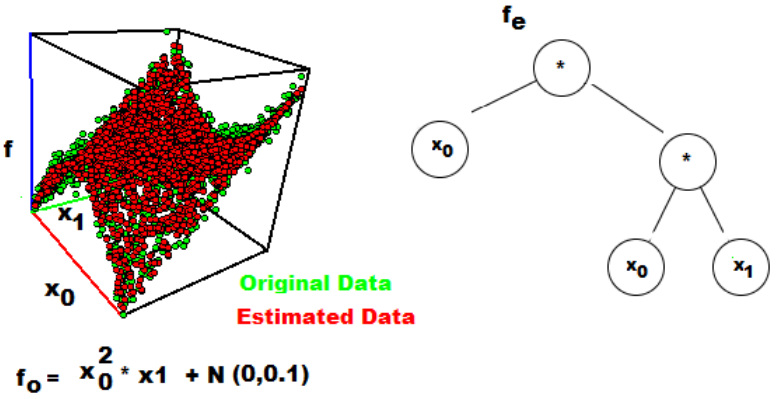


Figure 2.7: An example Symbolic Regression problem

The Santa Fe Trail problem is very different than the Symbolic Regression problem yet it can be solved using the same GP framework. The differences between this problem and the Symbolic Regression problem are the *functional symbols* that make up the expression trees and the definition of the fitness function. In this problem, a grid structured 2D map represents a field where 89 food pellets are positioned to form a trail. In this trail there are also a number of obstacles. An artificial ant starts from the top left position with a goal of following a trail and eating as much food as possible. The ant is expected to evade the obstacles and follow the food trail (figure 2.8).

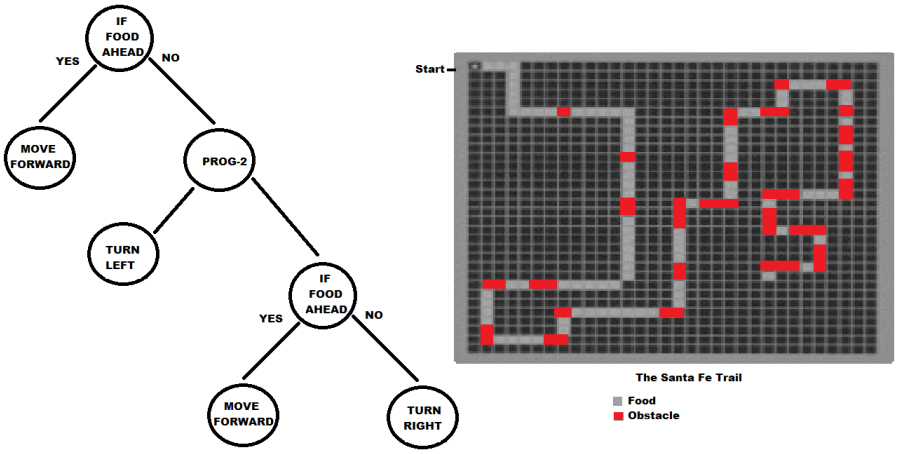


Figure 2.8: The Santa Fe Trail problem

The functional symbols for this problem are: If Food Ahead, Move Right and Move Left, Prog-2 and Prog-3. Prog-2 and Prog-3 functional symbols represent tree nodes with 2 or 3 child nodes and when these nodes are encountered the actions stated by the child nodes are taken starting from left to right. An example expression tree is also given in figure 2.8. The goal of GP is then to evolve an expression tree which lays out the action sequence for the ant to eat all 89 food pellets. The fitness function is simply the number of pellets left to be eaten which is to be minimized. During the evolutionary process, the algorithm learns a winning trait: 'if there is food ahead move forward'. Because this action increases the fitness, it is preferable for it to be included in the expression trees.

Determination of the functional symbols and the fitness function are problem specific steps one needs to take before using GP. The standard GP algorithm itself is domain independent and follows a certain flow as shown in algorithm 1.

Algorithm 1: Workflow of a standard GP system

```
//Initialization Randomly create an initial population of N individuals
repeat
    //Evaluation
    foreach individual in population do
        compute fitness of the individual
    end
    repeat
        //Mate selection
        Select individuals from the population as parents
        //Breeding
        Perform breeding:
        crossover,
        reproduction,
        mutation to generate new offsprings and replace parents
    until all individuals in the population are replaced with offsprings
until maximum number of generations are reached or an ideal individual is found
```

Though a tree-based representation is the earliest and the most commonly used option, alternatives such as the linear (vectoral) and graph based representations have also been proposed [117].

At the beginning of the GP process, the initial population is filled with randomly generated individuals. The size of the population is generally a user specified parameter which can also be determined dynamically during the run. There are various methods for initialization. The initialization process creates syntactically correct expression trees by combining the functional symbols. The two simplest methods are the *full* and *grow* operations. The full method generates full trees where all leaves are at the same

depth which is defined by the maximum tree depth parameter. The grow method creates trees with varying sizes and shapes up to the maximum tree depth. The *ramped-half-and-half* method that was proposed by Koza [83] combines these two methods in order to create a diverse set of initial expression trees. Each method is used to create half of initial population. The 'ramp' is a range of values for the maximum allowed depth to be used in the full and grow methods. Each method picks a random number within the range and creates the trees accordingly.

The rest of the evolutionary process is divided into generations in which a sequence of steps are taken to morph the initial population into a (hopefully) more optimal population with respect to the fitness function. The first step within a generation is the evaluation of each individual with respect to the fitness function. The selection step determines which individuals are going to be involved in breeding to create the individuals for the next generation. The simplest selection scheme is the roulette-wheel or the fitness proportional selection method. In this scheme, individuals with better fitness have higher probability to get selected. A variant of this technique that is known as the stochastic universal sampling (SUS) ensures that fit individuals are never left out by the selection [15]. These methods have a shortcoming that individuals with very close fitness values will have the same probability of getting selected which causes this method to act like random selection. The tournament selection method overcomes this shortcoming by considering the rankings rather than the proportions. This method prevents one extremely fit individual from invading the whole population and ensures that better individual is selected even in the case of very close fitness values. The tournament selection method is very simple. It randomly selects t individuals from the population in random (with replacement) and determines the best individuals within this group. The parameter t is known as the tournament size which controls the selection pressure. t=1 is random search with the lowest selection pressure and as the tournament size gets larger, the selection pressure gets higher, meaning that the algorithm tends to choose the fittest individual in the population. Lower selection pressure gives more chance to the weaker individuals to participate in breeding (see [96] for more details on these selection algorithms).

After the selection stage, new individuals are created through the breeding operators. The repro-

duction operator just clones the individual while the mutation operator modifies the expression tree of the individual. Reproduction and mutation operators take one individual and create one new individual and replace the selected individual with the new individual. The crossover operator hybridizes two individuals (parents) and produces two new individuals (offsprings) which is akin to mating in nature. By combining two individuals with high fitness, it is expected that an even fitter individual will be created. After crossover, each parent individual is replaced by one offspring. The selection and breeding cycle continues until the population is filled with new individuals. Elitism is a slight modification to this scheme, where a number of best individuals in the population are directly moved to the next generation without being subject to selection or breeding. The elitism strategy guarantees that the best fitness encountered during the evolutionary process never decreases.

Examples of crossover and mutation operations are shown in figure 2.9. In its simplest form (subtree crossover), crossover takes two individuals, randomly selects a subtree on each individual and swaps them. As per Koza's suggestion, the inner nodes have a higher probability of getting selected (%90) compared to the leaves (%10).

The subtree mutation operator randomly selects a subtree and replaces it with a randomly generated tree. Both operators respect the maximum tree depth limit. The mutate-swap mutation operator randomly selects two subtrees within the expression tree and swaps them. A large number of variations for crossover and mutation have been proposed in GP literature. More information and comparisons of these methods can be found in [117] and [96].

The standard crossover and mutation operators act on the genotype (the expression trees) and they are not guaranteed to effect the fitness in a positive way which is computed on the phenotype. It is the selection operator's job to match the fittest individuals so that the probability of creating a better individual is increased. However, the reproduction operators do not directly aim to increase fitness. Johnson points out this issue in [74] and urges the community to focus on *fitness-increasing crossovers* by concentrating on the semantics (the behavior of the programs) rather than the syntax (genotype). The semantically driven crossover (SDC) by Beadle and Johnson [16], the context aware

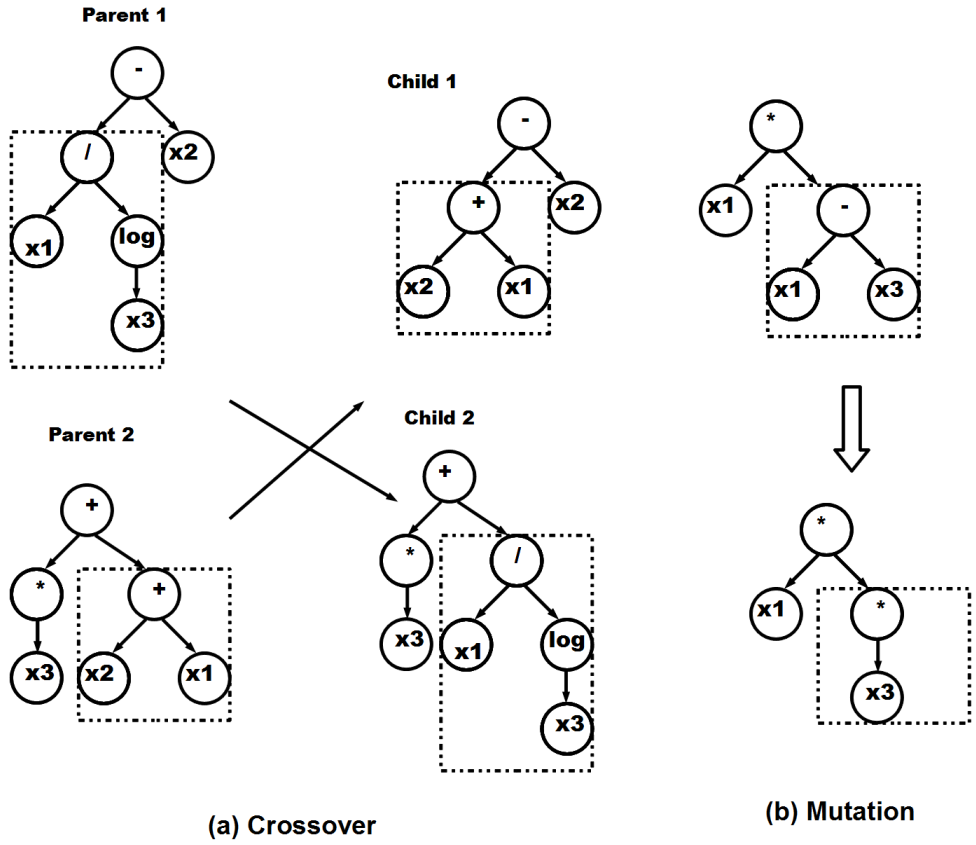


Figure 2.9: Crossover and mutation in GP

crossover by Majeed and Ryan [100], the semantics aware crossover (SAC) and semantic similarity-based crossover (SSC) methods by Uy et al. [149] and the functional crossover by Bongard [26] are techniques that consider the phenotype in crossover. Likewise, context aware mutation by Majeed and Ryan [101] and the semantically driven mutation by Beadle and Ryan [17] are GP mutation operators that focus on program behavior.

The GP system that was outlined in algorithm 1 is known as the 'generational genetic programming' which means that the whole population is renewed at each time step (generation). An alternative to this approach is known as the 'steady state genetic programming'. In this scheme, at each iteration a small number of new individuals (n) are created and the same number of pre-existing individuals are removed from the population so that the population size is kept fixed. For large number of n, this algorithm works similar to its generational counterpart.

2.4.2 Theory of Genetic Programming

Genetic programming is a nondeterministic algorithm, it might return drastically different results from one run to another, even failing badly in one run and then finding a perfect solution in the next. There are a lot of questions centered around why genetic programming works and what issues could be behind the failed cases. As it is stated in [119], since the early days of GP, a number of attempts have been made to devise a theoretical model which would explain the workings of GP. However, this goal has not been fully reached. Full theoretical analysis of simple GP systems have been reported but there are too many variations of the algorithm such as using different population initialization methods, different selection and breeding operators which would dramatically change the dynamics of the basic algorithm.

Fitness Landscapes A search landscape is a visualization of the search space and it has been widely used in characterizing problem difficulty. However, the landscape metaphor has been found to be difficult to model GP-based search. It is suggested that the complexity of the fitness landscape is not clearly related to the algorithm performance [89].

Markov Chains GP can be seen as a Markovian process since the next generation most likely depends on the current population. Roughly speaking, the whole population can be modeled as a point in some space that can be defined based on certain syntactical characteristics of the population and the trajectory that is generated during the evolutionary process can be examined. These models can be very complex and hard to build since the number of possible states that a population can shift towards is very large. Limited number of studies on certain GP configurations have been reported so far [118].

Schema Theorems These models are based on the idea of dividing the search space into syntactical subsets which are known as the 'schemata'. A schema theorem is concerned with the distribution of the population over the schema and how the distribution changes from one generation to the next. An 'exact' schema theorem can predict the distribution of the population over the schema accurately based on the information at the current generation [119]. Most of the earliest attempts for devising a schema

theorem only led to the development of lower bound estimations and an exact schema theorem was introduced by Poli et al. in [115]. This theorem considers GP with subtree crossover, the authors later extended the theorem to various other crossover operators.

Search Spaces The schema theorems and Markov chains models consider only the syntactical properties of the individuals along with the genetic operators but fitness is not taken into account. The investigation of search spaces is another strand of theoretical analysis of GP which is interested in studying the change of fitness distributions through the evolutionary process. The major outcome in this line of research is the finding that beyond a certain program size, the distribution of fitness values stays constant (see [89]) which can be interpreted as: "for each unique fitness value, there are more long programs than short programs".

2.4.3 Implementation Aspects of Genetic Programming

In this section, we discuss various aspects of genetic programming that would affect its success on problem solving.

2.4.3.1 Configuring the Runs

Genetic programming requires a large number of parameters and settings including the population size, number of generations, the minimum and maximum initial tree depth, maximum allowed tree depth, functional symbols, probabilities for each reproduction operators, tournament size for the selection operator. Eiben et al. discriminates between parameter tuning and parameter control [51]. In parameter tuning, the user is expected to determine the values for the parameters and perform multiple runs in order to determine the optimal values for their problem domain. In parameter control, the evolutionary process starts with initial parameter values and dynamically modifies the values during the run. De Jong gives an overview of the parameter setting problem in [43] and concludes that the goal of *parameterless* evolutionary algorithms is still far away.

2.4.3.2 Code Growth/Bloat

Bloat is an inherent problem in evolutionary optimization techniques that rely on a variable-length representation. In the context of genetic programming which uses a tree-based representation, the bloating phenomenon causes the search process to shift towards the space of larger expression trees *without significant improvement in fitness*. The side-effects of bloating are:

- The computational burden increases proportional to tree size,
- The larger the expressions are, less readible/comprehensible they get for the humans,
- Bloat and over-fitting were observed together in some GP problems. However, the relationship is not conclusive [110].

There have been a number of attempts to explain why bloating occurs. The earliest efforts have evolved around the term 'intron'. The most common definition of 'intron' is "redundant piece of code in the expression which can be removed without altering the behavior of the program". The 'hitchhiking' theory claims that introns are propagated via the crossover operator. The 'protection against crossover' theory claims that the introns that are present in the high fitness programs protect the individual from being disrupted by crossover by introducing more crossover points which are likely to preserve the 'useful code'. The removal bias theory suggests that in order to preserve fitness, the subtree that is removed from a tree should not be larger than the inactive code otherwise part of the 'valuable' code will also be cut off, however inserting a larger subtree with possible introns is not likely to disrupt the individual's fitness. Contrary to the common belief that "introns cause bloat", based on empirical evidence, Luke suggests otherwise in [95]. He reports that even when crossover is restricted in such as way that the introns are not allowed to be selected as the crossover points, the tree size continues to grow. Therefore, he suggests that introns appear because of tree growth not the other way around. The major non-intron theory is based on the 'program search space analysis' presented by Langdon and Poli in [89]. Based on this theory (see section 2.4.2), the authors state that for a given fitness value,

there are many more long programs than short programs, therefore, as generations pass, it is more likely for the GP to sample more of the longer programs.

A large array of methods have been proposed in GP literature in order to fight code bloat:

- Imposing depth and size limits on the expression trees [83],
- Editing/Simplifying the expressions [83],
- Incorporating size into fitness computation in a weighted linear combination scheme (parsimony pressure) [83],
- Bloat aware genetic operators such as size fair crossover [87] and mutation [86],
- Bloat aware selection operators such as the lexicographic tournament selection favoring the shorter individual if fitness values are equal [97],
- Population control by removing above-average sized individuals from the population (such as the Tarpeian method [116]),
- Operator equalization [46] and its variants which try to control the distribution of program sizes within the population can introduce bias towards either smaller or larger programs,
- Multi-objective methods with expression size as an explicit objective [23, 76],
- Order or nonlinearity as a complexity measure in symbolic regression problems [158]

Detailed information and comparisons on bloat control mechanisms can be found in [7, 96, 136].

2.4.3.3 Generalization Power/Over-fitting

Lack of generalization or over-fitting is an issue for every data backed learning algorithm. Highly over-fitting systems perform well on the training data but fail to generalize to unseen data. Kuschu studied the robustness of GP based solutions and suggested the use of test data to increase generalization power [85]. In GP based data analysis problems, a three data set (training, validation and test set)

methodology have been utilized by a number of researchers. Use of a separate test set prevents the researchers from reporting results that lack generalization. The validation set helps test the generalization power of the solution during the training process and aims to create more robust solutions [59]. In machine learning, over-fitting is related to the complexity of the solution. A complex solution would fit the noise in the data and therefore will not successfully generalize to unseen data. For this reason, bloating and over-fitting have been thought to be related issues in GP. However, there are mixed results on this topic. Bloating is measured in terms of stagnancy in fitness despite increasing syntactic complexity of the solutions. A number of attempts have also been made to study the relationships between the behavioral complexity and generalization power of GP solutions. A more detailed overview of over-fitting in GP is presented in section 6.5.

2.4.3.4 Population Diversity and Premature Convergence

In GP based systems, it has been widely observed that success rate varies from run to run. In some runs, the population converges to a less than fit solution and a better solution can not be found no matter how many more generations progress (premature convergence). Many researchers report the loss of diversity in the population as a cause for the failed runs. Burke et al. [31] study various ways to measure population diversity and their relationships to success of the runs. The authors utilize genotype based and phenotype based measures for diversity and report that successful results occur when population contains high number of individuals with similar structures but diverse fitness values.

A number of methods have been introduced in order to maintain and increase diversity. Multi-population approaches explicitly aim for diversity. Lineage selection [32] and No Same Mates approaches [63] promote diversity by allowing dissimilar individuals to participate in crossover. The Age-Layered Population Structure (ALPS) approach by Hornby [67] uses age to indicate the number generations for each individual in the population. The newly generated individuals inherit their parents' age as a way to keep track of genetic lineage. Another age-based method is proposed in [132] where the age of an individual is used as an explicit criterion in a multi-objective setting. Overviews on the concept

of diversity in genetic programming in general can be found in [31, 62]. A detailed discussion of how population diversity effects the success/failure of the runs in GP based data classification systems is given in section 6.3.

2.4.4 Multi-objective Genetic Programming

The standard GP that was introduced in the last section works by comparing scalar fitness values. In the case that there are multiple objectives to consider (such as accuracy versus expression tree size), a number of methods have been introduced so that the problem can still be solved within the framework of standard GP. Creating weighted linear combinations of the objectives or prioritizing the objectives such as in the lexicographic tournament selection are the main methods of this sort. In each case, the user needs to determine the weights (trade-offs between the objectives) or the priorities between the objectives.

In this section, we consider a more principled framework of handling multiple and possibly conflicting objectives which frees the user from having to determine the trade-offs or priorities between the objectives. We first introduce the concept of 'Pareto dominance' which is the foundation of evolutionary multi-objective (EMO) algorithms. We then discuss the main components of an EMO algorithm and present an overview of two of the most common EMO frameworks.

The term Pareto-dominance was named after Italian economist Vilfredo Pareto who introduced the concept within the context of income distribution. He suggested that "maximum economic satisfaction in a population can be reached when no one can be improved without degrading someone else." [113]. Pareto dominance is defined as follows:

Definition 1. I_X *dominates* I_Y if $\forall i, I_{X_i} \leq I_{Y_i}$ and $\exists k, I_{X_k} < I_{Y_k}$ where I_{X_i}, I_{Y_i} are the individual criteria. The set of all *non-dominated* solutions are called the pareto-optimal set.

Figure 2.10 shows a two-objective case where both objectives are to be minimized. The red dots indicate the non-dominated data points on the graph.

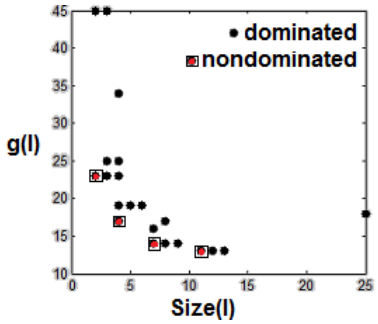


Figure 2.10: Pareto-dominance

Evolutionary computing algorithms are very suitable for multi-objective optimization since they generate multiple solutions (population) to a problem at any given time and selection of the 'best' solutions amongst them can be easily performed using the Pareto-dominance concept.

In an evolutionary computing setting such as GP, the goal is to find an approximation to the Pareto-optimal set (Pareto front) [1] of solutions to a given problem. An EMO algorithm has three goals: 1) guiding the search towards the Pareto-optimal solutions, 2) making sure that the solution set at each step is diverse and 3) ensuring that the non-dominated solutions do not get lost during the evolutionary process. The guidance of the search towards optimal solution set is achieved by assigning fitness values with respect to Pareto-dominance. The diversity preservation in the solution set is achieved using density information where an individual's probability of getting selected is reduced if it is in a highly condensed area of the multi-objective space. Preservation of the non-dominated individuals is achieved by incorporating an elitism mechanism through an auxiliary storage which is called the 'archive'. Based on these principles, various EMO algorithms have been proposed in evolutionary computing literature.

The NSGA-II (Non-dominated Sorting Genetic Algorithm II) was introduced by Deb et al. in [42]. This algorithm separates the population into multiple layers of Pareto-fronts. A custom tournament selection operator selects the individuals for breeding. Within a tournament, the individual from the lowest front is selected (minimization of the objectives is assumed here). The ties are broken with respect to the 'crowding distance' which is a measure of density based on the sum of distances between an individual

[1] In the general multi-objective optimization setting, the Pareto set and Pareto front are two different things. Pareto set is defined on the decision space (e.g. genotype) and Pareto front is defined by the objective space (e.g. phenotype). The mapping from the decision space to the objective space is a surjective function; multiple genotypes can have the same fitness vector. The algorithms presented here consider the objective space.

an its nearest neighbors within the same Pareto-front layer. Elitism is performed through considering the whole population of parent and offspring individuals after breeding and re-computing the layers of Pareto-front and then discarding the worse half of the individuals.

The SPEA2 (Strength Pareto Evolutionary Algorithm 2) framework [165] was developed by Zitzler et al. The algorithm maintains a fixed sized 'archive' which holds the current Pareto-front. At each generation, a two-component fitness assignment is performed. The raw fitness component of the individual's fitness is computed with respect to the number of individuals it dominates and the number of individuals that are dominated by it in the whole population. When there are many non-dominating individuals, the raw fitness will not be a good measure for selecting individuals for breeding. Therefore, a second component is added to fitness computation. A nearest neighbors algorithm based density estimate for each individual is computed as the inverse of the distance of the individual to its k-the nearest neighbor in the population. A small distance indicates that the individual is located in a densely populated area, thus taking the inverse of this measure increases the density component of the fitness and potentially marks this individual as a rather unfavorable one for selection. The final fitness of the individual is then the sum of the raw and density fitness values. After breeding takes place, the archive is re-built with the non-dominated individuals. Most frequently, the archive is either not filled up completely or over-filled. In the first case, a number of dominated individuals are allowed to enter the archive until it is filled. Otherwise, the archive is truncated. Truncation is done in a manner that the individuals from the highest density areas are removed from the archive in order to preserve diversity.

The ParetoGP algorithm introduced by Smits and Kotanchek [142] maintains two separate populations that are both randomly initialized. One population is called the archive. The algorithm performs two separate Pareto tournaments for the population and the archive. For each tournament, only the non-dominated individuals are considered. The crossover operator creates new individuals by hybridizing one randomly selected tournament winner from the population and one randomly selected tournament winner from the archive. At each generation, the population is replaced by these newly generated individuals. The process is repeated for a number of generations which is called *cascades* by the authors.

Then, the non-dominated individuals in the current population are moved to the archive and the archive is truncated if necessary. A new cascade then starts by re-initializing the population. The algorithm controls the exploration aspect by resetting the population periodically and the archive controls the exploitation aspect.

Java language implementations of both methods are available in the ECJ framework [94] and ParetoGP is included in a commercial data modeling package [1]. Surveys and comparisons of the various EMO frameworks can be found in [41, 37].

2.5 Datasets

A number of datasets are used throughout this thesis for experiments. In this section, we provide additional information about each dataset beyond how many attributes and groups they contain. The majority of the datasets were downloaded from the UCI repository [3] unless otherwise stated. These datasets were selected because of their common use in machine learning and also in genetic programming literature. Some datasets contained missing records and/or unnecessary attributes. We also explain any data cleansing step we followed. Unless otherwise noted, no other pre-processing such as normalization or standardization have been applied to the datasets.

Pitch Accent Detection: This dataset was generated by Dr. Andrew Rosenberg of Queens College based on the Boston Directions speech dataset. The task is to detect pitch accents in spoken American English which are characterized by excursions in pitch, intensity, duration and increased high-frequency emphasis identified with spectral tilt [127]. Originally, 200 features were generated from speech wave files using a toolkit named AutoBI developed by Dr. Rosenberg. Each record is labeled as accented or de-accented. A sample of 2000 records (1000 samples for each class) were selected for analysis and the number of features were reduced to 45 using the Relief [81](non-wrapper method) attribute selection algorithm (with 10-fold cross-validation) on Weka.

BUPA Liver Disorders: This dataset contains five blood test results for a group of male patients. These tests are thought to be related to liver disorders possibly caused by excessive drinking. The attributes are: mean corpuscular volume (MCV), alkaline phosphotase (ALKPHOS), alamine aminotransferase (SGPT), aspartate aminotransferase (SGOT), gamma-glutamyl transpeptidase (GAMMAGT), number of half-pint equivalents of alcoholic beverages drunk per day (DRINKS). The dataset contains 345 records where each record is labeled as either positive or negative for liver disorder.

Leptograpsus Crabs: This dataset contains five physical measurements for 50 speciments of each gender of two species (identified as Orange and Blue) of rock crabs. The measurements are: the width of the frontal lip (FL), the rear width RW, the length along the midline CL, the maximum width of carapace (CW) and the body depth (BD) all in millimeters [123]. Each row is labeled as Male/Orange, Female/Orange, Male/Blue and Female/Blue.

PIMA Indians Diabetes: The data was collected by the US National Institute of Diabetes and Digestive and Kidney Diseases. It contains samples from a diabetes test that was administrated to a group of Pima Indian women of at least 21 years old living in Arizona. The attributes of the dataset are: number of pregnancies (TP), plasma glucose concentration in an oral glucose tolerance test (PG), diastolic blood pressure (BP), triceps skin fold thickness (FT), serum insulin (SI), body mass index (BM), diabetes pedigree function (DP), age in years. The original dataset contains 768 records. Here, we are using a sanitized version of the dataset as suggested in [123]. The attribute SI was removed due to high number of missing values. Then, any record with a missing value has been purged, leaving 532 complete records. Each record is labeled as either diabetic or non-diabetic.

Johns Hopkins University Ionosphere: This dataset contains 351 radar readings from a system consisting of an array of 16 high frequency antennas. The systems targets free electrons in the ionosphere and labels each radar return as *good* or *bad* depending on evidence of some structure in the ionosphere. Bad returns do not show any structure since the signals pass through the ionosphere. The

received signals were further processed into 34 continuous attributes. The attributes are numbered from 1 to 34. After inspection, we removed attribute 2 since it had all 0 values.

Olive Oils: This dataset contains the amounts of eight fatty acids characterizing 572 olive oils from nine regions of Italy. This dataset was introduced in [166] and downloaded from [2].

Segments: This is an image recognition dataset that was included in Weka distribution [4]. This version contains 1500 records and 19 continuous features representing a 3x3 patches from a set of ourdoor images. Each image is hand-segmented and each segment is assigned a label according to scene type (brickface, sky, foliage, cement, window, path, grass). The feature set contains the mean and standard deviation of such attributes as line density, contrast and color intensity for each 3x3 patch. Upon inspection, we removed one attribute since it was simply the total number of pixels per patch which was 9 for all records.

Wisconsin Breast Cancer (WBC): This dataset contains 9 attributes that are related to identifying breast cancer: clump thickness (CT), uniformity of cell size (UCSZ), uniformity of cell size (UCSZ), uniformity of cell shape (UCSH), marginal adhesion (MA), single epithelial cell size (ECSZ), bare nuclei (BN), bland chromatin (BC), normal nucleoli (NN) and mitoses (MT). The original dataset downloaded from the repository contains one extra attribute that was found to be the record number and consequently removed. After elimination of the rows with missing values, the final dataset we use here contains 683 records. Each record is labeled as either benign or malignant.

Wisconsin Diagnostic Breast Cancer (WDBC): The attributes of this dataset were generated from a set of 569 digitized images of fine needle aspirate (FNA) of a breast mass. Each image is labeled as malignant or benign. For each image, ten measurements that characterize the cell nuclei were derived and the mean (M), standard deviation (SD) and the worse (W) values of each measurement were computed for a total of 30 attributes: radius (MR,SDR,WR), texture (MT,SDT,WT), perimeter

(MP,SDP,WP), area (MA,SDA,WA), smoothness (MSM,SDSM,WSM), compactness (MCT,SDCT,WCT), concavity (MCO,SDCO,WCO), concave points (MCP,SDCP,WCP), symmetry (MSY,SDSY,WSY), fractal dimension (MF,SDF,WF).

Wine: The dataset contains the chemical analysis results on 178 wine samples of three different kinds from the same region of Italy. The thirteen attributes are: alcohol (ALC), malic acid (MACD), ash (ASH), alcalinity of ash (AASH), magnesium (MG), total phenols (TPEH), flavanoids (FLV), nonflavanoid phenols (NFLV), Proanthocyanins (PROA), Color intensity (CINT), hue (HUE), OD280/OD315 of diluted wines (DIL), proline (PROL).

Dataset	# **records**	# **variables**	# **classes**
Accenting	2000	45	2
BUPA	345	6	2
Crabs	200	5	4
Ionosphere	351	33	2
Olive Oil	572	8	9
PIMA	532	7	2
Segment	1500	16	8
WBC	683	9	2
WDBC	569	30	2
Wine	178	13	3

Table 2.3: Properties of the datasets

Chapter 3

Human Perception and Automated Measures of Interpretability of Data Transformations

Visual Analytics is a human-machine collaboration in decision making based on observed data. In most cases, it is impractical for humans to manually explore the large amounts of data to discover hidden patterns that would be useful in decision making. Therefore, the duty of the machine is to provide automated data analysis in order to help the human-the decision maker. On the other hand, humans need to be able to interpret and comprehend the machine generated models so that they can make informed decisions.

The best way to make sure that the machine will return interpretable models is to provide instructions to search for such models. Since this thesis is focused on the classification problems, we study interpretability within the framework of exploratory analysis of labeled data through visualizations. Two-dimensional scatterplots are amongst the most common types visualizations since the early days of statistical graphs. Here, we consider scatterplots of higher dimensional datasets with axes generated

by feature transformation functions mapping the original features (variables, attributes) into the two projected features.

We consider two important aspects of human interpretability of data projections for labeled data. *Visual interpretability* is concerned with how humans judge the quality of the views presented as 2D scatterplots. In the case of labeled data, visual interpretability closely relates to how easy it is to tell the members of each class apart by inspecting the scatterplot. *Semantic interpretability* is concerned with how the views are generated, namely the complexity of the projection functions that relate the projection axes to the original variables. A number of automated measures from machine learning and visual analytics can be used to assess visual interpretability. The goal of our study was to investigate the degree of which the automated measures of visual and semantic interpretability match human perception.

We designed a two-part experiment to study visual and semantic interpretability. In the first part, we showed the participants a number of scatterplots and asked them to rate these views. The participants were not given any background information about the data since we aimed to investigate how they would rate the views independent from a specific domain. The second part of our experiment investigated how easily humans would understand mathematical expressions with varying levels of complexity. We used a generic set of expressions to study interpretability independently from a specific domain.

We chose to separate the evaluation of these two interrelated aspects of how users might judge the quality of data projections rather than showing the visualizations along with the mathematical expressions transforming the original dataset into the displayed view. Such an experiment setting would impose another level of complexity to our study. The user judgement on visualizations would be affected by how they feel about the corresponding transformation functions and vice versa. Therefore, it would be difficult to be isolate how they would rate the visualization or the transformation function and compare with the corresponding automatic measures. Since the goal of this thesis work is to provide automatically generated data projections that are both visually and semantically interpretable to the user, we needed to study these aspects in isolation. Thus, learning the subjective visual/semantic

interpretability trade-off is beyond the scope of this thesis work.

3.1 Previous Work

A number of authors have previously conducted user studies in order to compare various automated measures to human perception. The VizRank algorithm proposed by Leban et al. searches for *informational* 2D projections of datasets that are evaluated by a k-Nearest Neighbor classifier [90]. The authors claim *almost perfect* agreement between the human judgement and the VizRank algorithm through a user study conducted using six datasets. Sips et al. propose two measures based on the notion of *class consistency* in [137]. One measure is based on preservation of closeness to class centroids after projection, and another is based on the entropies of the spatial distributions of the classes. The authors report a user study on a number of datasets with varying number of classes. They claim that their proposed measures are in alignment with human judgement in terms of finding all views that were labeled as good views by the participants. Tatu et al. propose two measures to evaluate the degree of separation on scatterplots of labeled data in [143]. Tatu et al. report a user study in [144] that compares four visual quality measures that have been proposed in [137] and [143]. The authors suggest that a combination of measures might be worth investigating.

As opposed to the various studies on visual interpretability or quality of the projections of labeled data, the interpretability of the projection axes have been addressed only in a few studies. In the case of projection pursuit, the projection functions are given as weighted linear combinations of the original features. Morton defines the interpretability of these projection functions in terms of parsimony (simplicity) and proposes rotation and entropy based methods to simplify the coefficients of the linear projections while preserving the *interesting* view [105]. El-Arini et al. present a dimensionality reduction method that searches over scatterplots generated by simple arithmetic expressions of the original features and assessed by accuracy of a Bayesian classifier [80]. The authors claim that expressions containing more than one or two features become less interpretable. However, no empirical study has

been reported to justify this intuition.

There have been a number of related studies in the Human Computer Interaction (HCI) field investigating how humans perceive the complexity of mathematical expressions in order to develop improved human-computer interfaces. Anthony et al. study the effects of different input methods (keyboard, handwriting, speech) with respect to the complexity of the mathematical expressions for the purpose of developing intelligent tutoring systems [11] for algebra. The study presented by Awde et al. in [13] aims to find the most appropriate way to present a mathematical expression to visually impaired users. The modality of the presentation is selected based on a notion of human perceived complexity of mathematical expressions inferred through a user study. Their experimental design is similar to ours, where the participants are shown a number of mathematical expressions and asked to re-write and rate the expressions. Then, a relationship between the structural properties of the expressions and the human perceived complexity is derived. The set of expressions they chose came from a wide range of fields including logic and calculus. In our experiments, the set of expressions were limited to various linear/non-linear combinations of a limited number of variables representing possible data projection functions of varying complexity.

The rest of this chapter presents our experiments on interpretability of data projections along with a study of how the automated measures relate to human perception.

3.2 User Study on Interpretability of Visualizations

We developed computer software that automatically administered the data visualization experiment without investigator intervention. In this section, we present the details on design and execution of the study along with our findings on how well the automated measures match the human perception of visual interpretability.

3.2.1 Participants

We recruited 20 participants (13 males and 7 females) who had completed or were pursuing graduate degrees in scientific fields such as computer science, physics, biology, engineering, accounting and psychology.

At the beginning of the study, the participants were asked to fill out a brief questionnaire asking them about their related course-work or experience. 14 of the participants specified that they had taken a Statistics, Data Mining or a Machine Learning course.

3.2.2 Datasets and Visual Interpretability Measures

We chose four commonly used datasets from the data mining and visual analytics literature (table 3.1). These datasets were selected because they contain different number of classes ranging from 2 to 9, which would let us investigate how the number and shape of the classes affect the relationship between human perception and the automated measures.

The Wisconsin Diagnostic Breast Cancer (WDBC) dataset contains 30 measurements characterizing malignant or benign tumors. The Wine dataset contains 13 attributes related to the chemical properties of wines from three different regions of Italy. The Segments dataset contains 19 features derived from images of seven kinds of scenes (brickface, sky, foliage, cement, window, path, grass). All three datasets were downloaded from the UCI Machine Learning Repository [3]. The Italian olive oils dataset [166] contains the amounts of eight fatty acids in olive oils that are from nine different regions of the country (downloaded from [2]). The Wine and Olive oils datasets have also been used in previous user studies mentioned at the beginning of this chapter.

Name	**#features**	**# classes**	**# scatterplots**
Wisconsin Diagnostic Breast Cancer (WDBC) [3]	30	2	435
Wine [3]	13	3	78
Segment [3]	19	7	171
Italian olive oils	8	9	28

Table 3.1: Datasets

In order to compare human judgement on quality of 2D views of labeled data to automated measures

we chose a number of measures from various fields of machine learning and visual analytics. Wrapper based methods from the feature extraction field have been designed to assess the usefulness of the extracted features with respect to the performance of classification algorithms. In this user study, we utilized a number of commonly used classification algorithms to assess the quality of the scatterplots for the four datasets mentioned above (section 2.3.1.1). The Naive Bayes, K-Nearest Neighbors, Support Vector Machine and Decision Tree algorithms are amongst the most common 10 machine learning algorithms according to [161]. Our goal is to investigate how each algorithm's decision boundary characteristics would relate to human perception of class separation on different datasets with varying number of classes.

A number of cluster validity indices have been proposed in data clustering literature in order to assess the quality of groupings generated by different clustering algorithms. These indices can also be used to measure the quality of the groupings on 2D scatterplots with respect to the class labels of the observations. For our experiments, we included three cluster validity indices (section 2.3.1.2). A number of visual quality measures have been introduced in visual analytics literature ([91, 137, 143]). We included two of the proposed measures that were reported in [144] as the closest matches to the human perception through user studies (section 2.3.1.3). Table 3.2 presents the list of the automated measures we utilized in our experiments.

Name	**section**
Naive Bayes [4]	2.3.1.1
Radial Basis Function (RBF) Classifier [4]	2.3.1.1
k-Nearest Neighbors (k-NN) [4]	2.3.1.1
Support Vector Machine (SMO) [4]	2.3.1.1
Logistic Regression Classifier [4]	2.3.1.1
C4.5 Decision Tree [4]	2.3.1.1
Simple CART Decision Tree [4]	2.3.1.1
C Index (I_C) [68]	2.3.1.2
Davies-Bouldin Index (I_{DB}) [39]	2.3.1.2
Dunn Index (I_{Dunn}) [48]	2.3.1.2
LDA Index (I_{LDA}) [91]	2.3.1.3
Class Consistency Measure (CCM) [137]	2.3.1.3
2D Histogram Density Measure (2D-HDM) [143], 100x100 bins	2.3.1.3

Table 3.2: Visual interpretability measures

For a dataset with N attributes, there are $N(N-1)/2$ unique attribute pairs, where each pair can be visualized as a scatterplot. In order to choose the visualizations for our experiment, we first generated

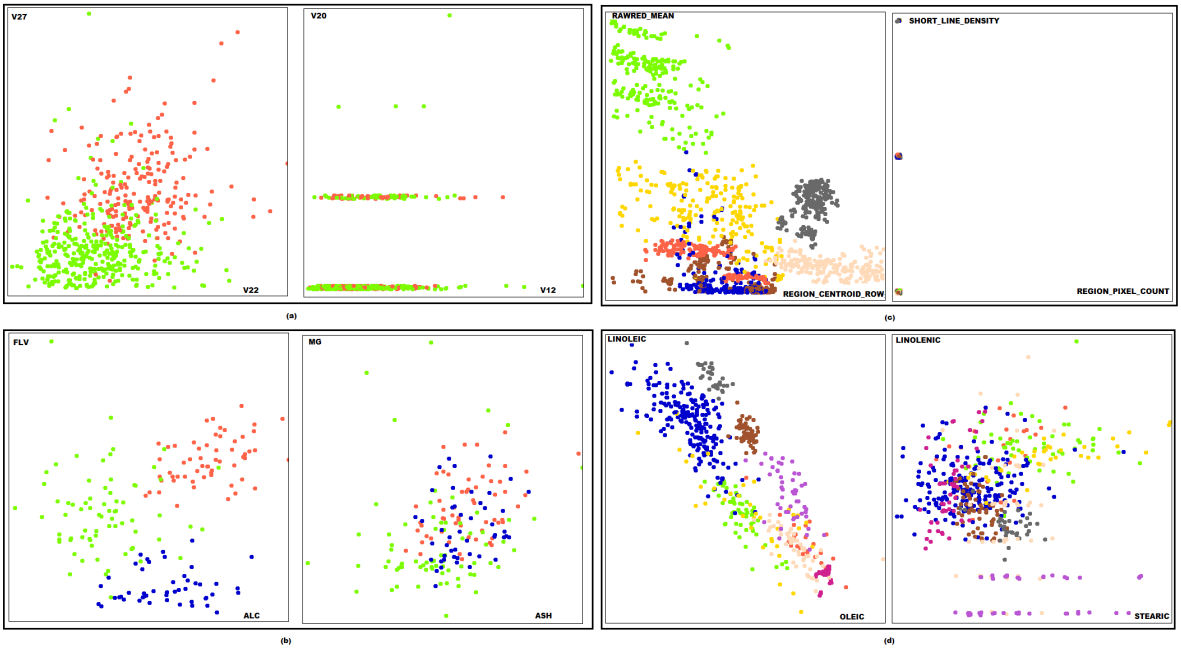


Figure 3.1: Best (left) and worst (right) views from the selected set of scatterplots based on the median value of all automated measures. (a)WDBC (b)Wine (c)Segment (d)Olive oils

all possible 2D scatterplots for each dataset. We computed the values of the automated measures given in table 3.2 for the datasets. Our aim was to ensure that we chose a diverse set of visualizations with respect to the automated measures. We created five equi-width bins of values between [0-1]. Then, for each bin, we selected two scatterplots that appeared most frequently within that value range across all the automated measures. Upon completion of this process, a total of 40 scatterplots (10 for each dataset) were selected to be included in our experiments.

3.2.3 Task

Before starting the study, the participants were told that they would be shown a series of scatterplots of some datasets containing multiple groups and their task was to rate how good the view was by inspecting the visualizations. We did not provide any directions on how to define the “goodness” of a view. Each scatterplot showed only the data points in different colors with respect to their class labels and no further information about the data (such as the name of the dataset or the names of the attributes) was provided. After reading the instructions, the participants were shown one scatterplot at

a time and were asked to rate them on a continuous scale between 0 (very good) to 1 (very bad) with labels shown on table 3.3.

Rating	Value
Very Good	0.0
Good	0.25
Average	0.5
Bad	0.75
Very Bad	1.0

Table 3.3: Labels for visual interpretability ratings on continuous scale

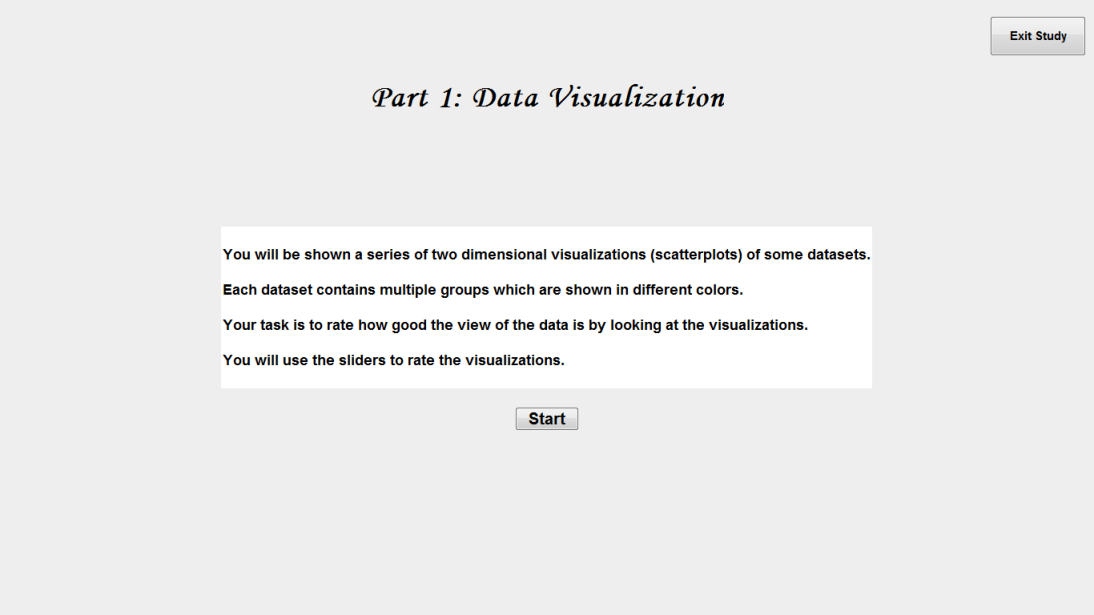
3.2.4 Methodology

Each participant rated a total of 45 scatterplots. Undisclosed to the participants, the first five scatterplots were artificial views showing different levels of compactness and separation between the classes from *very good* to *very bad* with respect to the automated measures. These visualizations were used as *calibration views* in order to help the participant get used to the interface and build their mental models for how they would rate the quality of a view. The participant ratings for these calibration views were not included in the analysis of the responses. The remaining pre-selected scatterplots were displayed in a randomized order to each participant. In order to reduce the effect of outliers, we computed the median of participant responses for each of the scatterplots and used this value in our comparisons to the automated measures.

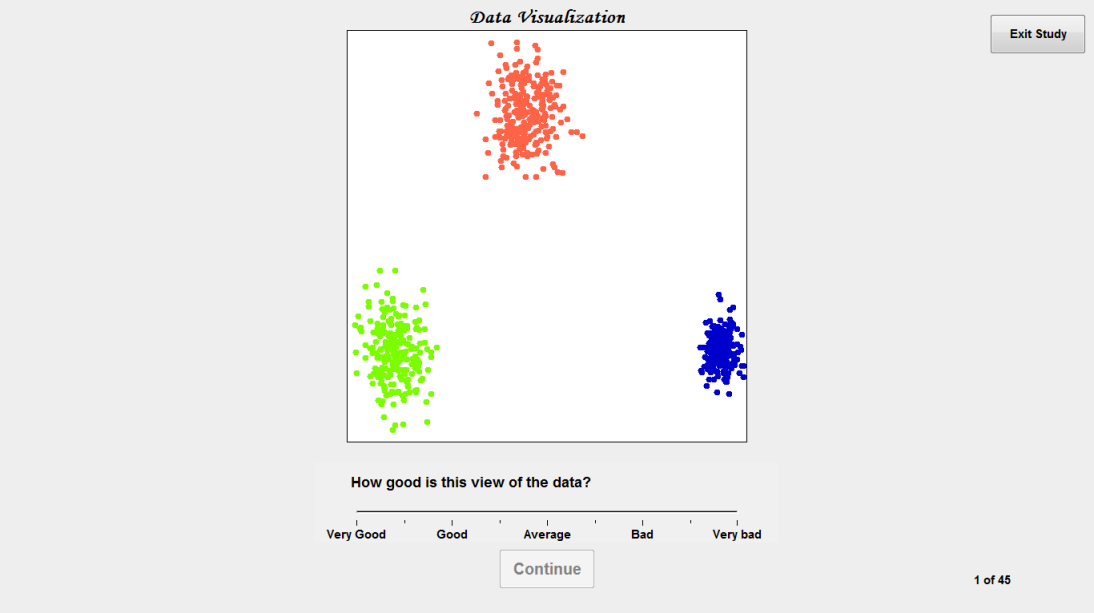
3.2.5 Results

Figures 3.3- 3.6 show boxplots for participant responses on views from each individual dataset. Each boxplot summarizes the distribution of participant responses for the image placed under the boxplot. Within each dataset, the boxplots are ordered from good (left) to bad (right) with respect to median of participant responses.

For some scatterplots, the range of human responses are wider than others, indicating some disagreement in judgement within the group of participants. The most controversial image seems to be scatterplot $\#5$ from the Wine dataset (figure 3.4). The view gives the impression that there are three



(a) Participant instructions page



(b) One calibration view

Figure 3.2: Visual interpretability user study screenshots

separate regions but there are also considerable amount of points that seem to have been misplaced.

In summary, the plots show that the participants typically rated the views showing separation between the groups as better views compared to those that had no clear separation.

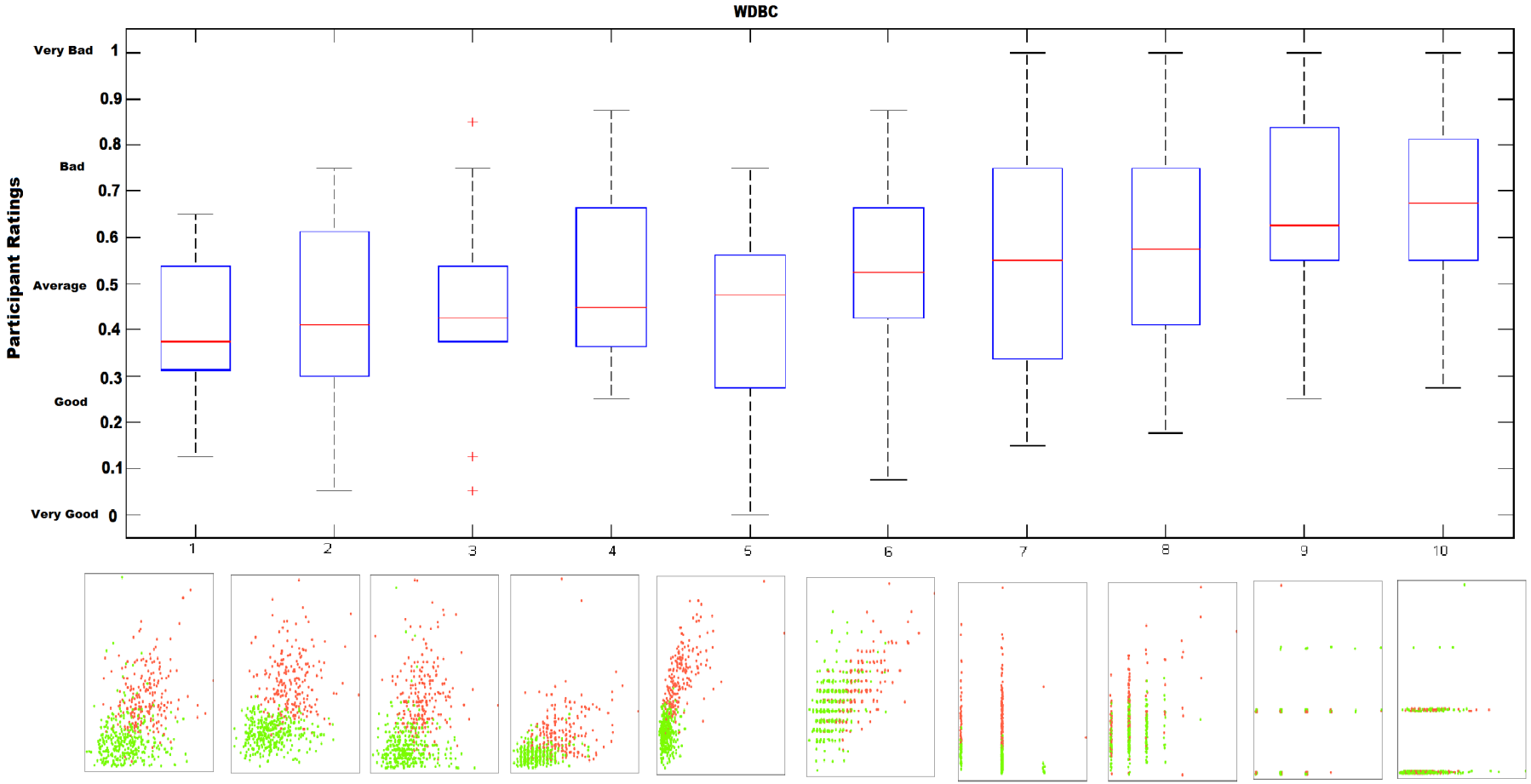


Figure 3.3: Summary of human responses for visualizations from the WDBC dataset

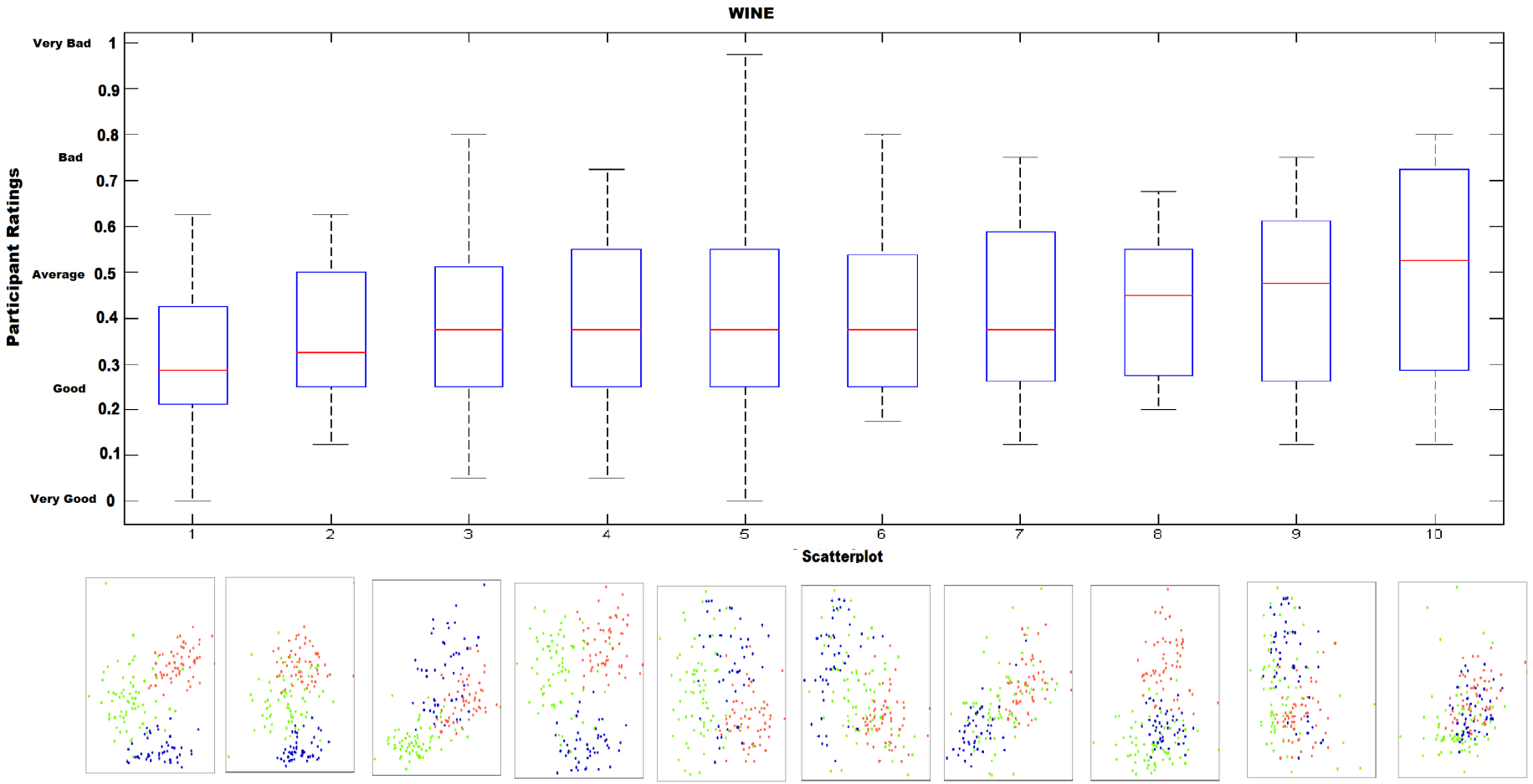


Figure 3.4: Summary of human responses for visualizations from the Wine dataset

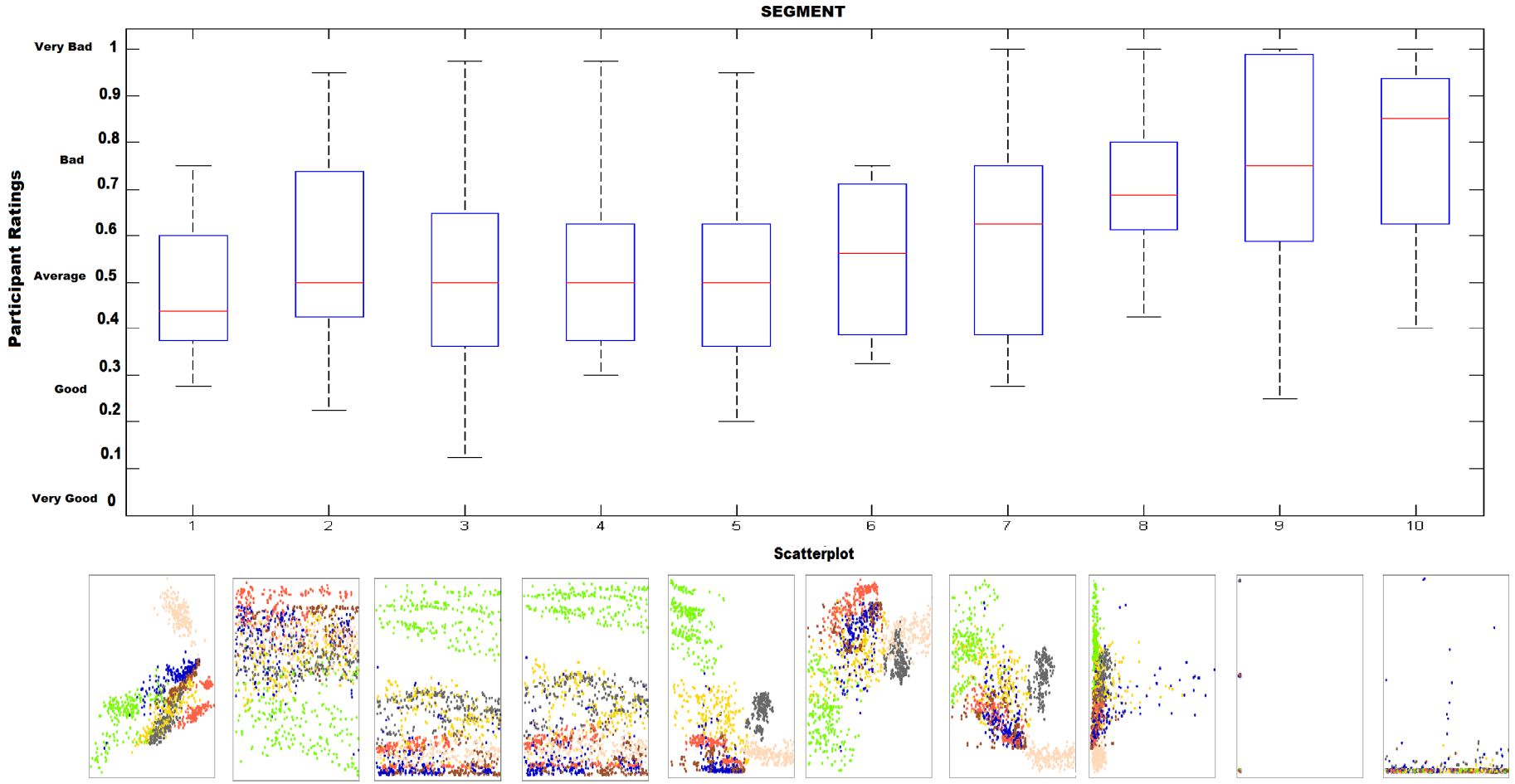


Figure 3.5: Summary of human responses for visualizations from the Segment dataset

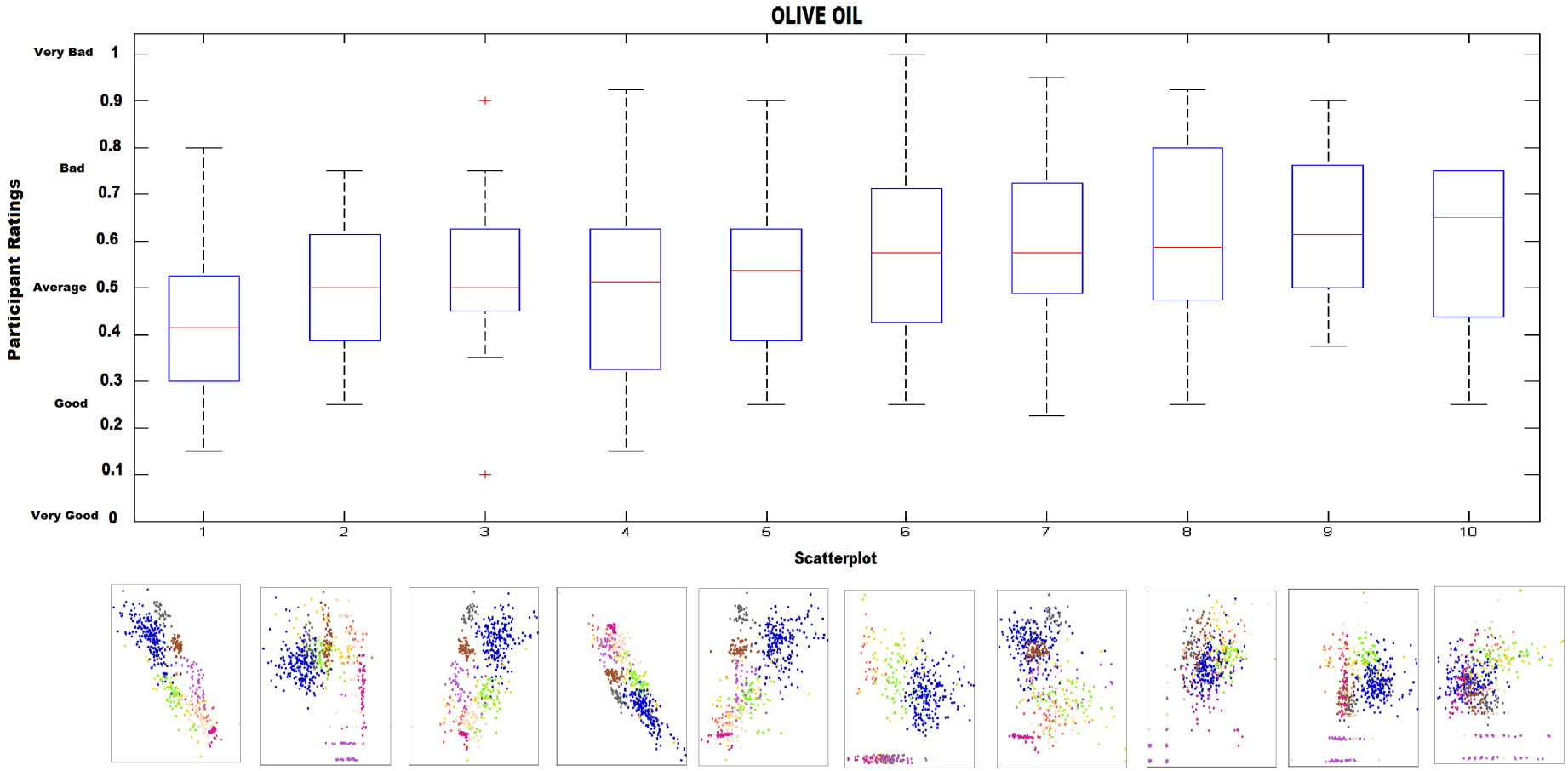


Figure 3.6: Summary of human responses for visualizations from the Olive oil dataset

Table 3.4 summarizes the relationships between each measure and human perception for all scatterplots[1]. The R^2 values indicate how well the linear regression model fits the data in terms of predicting the participant ratings based on each automated measure. Therefore, the measures are sorted in descending order with respect to the R^2.

Measure	**SSE**	R^2	**Adj.** R^2	**RMSE**	**p-value**
Support Vector Machine	0.2461	0.5506	0.5388	0.0805	< 0.05
Naive Bayes	0.2516	0.5406	0.5285	0.0814	< 0.05
Radial Basis Function Classifier	0.2577	0.5293	0.5169	0.0824	< 0.05
Class Consistency Measure	0.2603	0.5246	0.5121	0.0828	< 0.05
Logistic Regression Classifier	0.2635	0.5188	0.5061	0.0833	< 0.05
Dunn Index	0.2712	0.5047	0.4916	0.0845	< 0.05
K-Nearest Neighbors	0.2785	0.4914	0.4780	0.0856	< 0.05
Simple Cart Decision Tree	0.2951	0.4611	0.447	0.0881	< 0.05
C4.5 Decision Tree	0.3064	0.4404	0.4257	0.0898	< 0.05
Davies-Bouldin Index	0.3178	0.4197	0.4044	0.0914	< 0.05
2D-Histogram Density Measure	0.3258	0.4050	0.3893	0.0926	< 0.05
LDA Index	0.5112	0.0664	0.0419	0.1160	0.11
C Index	0.5314	0.0295	0.0040	0.1183	0.29

Table 3.4: Summary of linear relationships (Df:38, $\alpha = 0.05$) between the automated measures and human perception on all scatterplots

When we consider participant ratings for all scatterplots regardless of the dataset, we see that except for the LDA Index and the C Index, the correlation between human perception and each automated measure is statistically significant at $\alpha = 0.05$ level. Two classification algorithms, Support Vector Machine and Naive Bayes ranked the top two, tightly followed by the Class Consistency Measure, Dunn Index and K-Nearest Neighbors. Amongst the classifiers, the two decision tree based algorithms (J48 and Simple CART) were the bottom two in matching participant ratings.

Figure 3.7 presents the plots showing the relationships between human responses and the ratings generated by each classifier based automated measure. The rating for each classifier is computed as the 10-fold cross-validation estimate of the error rate on the 2D data corresponding to each individual scatterplot.

As the plots reveal, the human ratings over the 40 scatterplots range approximately from 0.3 to 0.85 while the classifier ratings range from approximately 0.1 to 0.85. Even though it is highly likely that humans and the classifiers can tell the difference between a good and a bad view, classifiers seem to be more generous in their ratings of good views. This observation calls for a closer attention. It is

[1]The goodness-of-fit measures: SSE (Sum of Squares of Error), RMSE (Root Mean Squared Error), Adjusted R^2, p-value: p-values computed by the two-tailed t-test (H_0: Pearson's correlation is not 0)

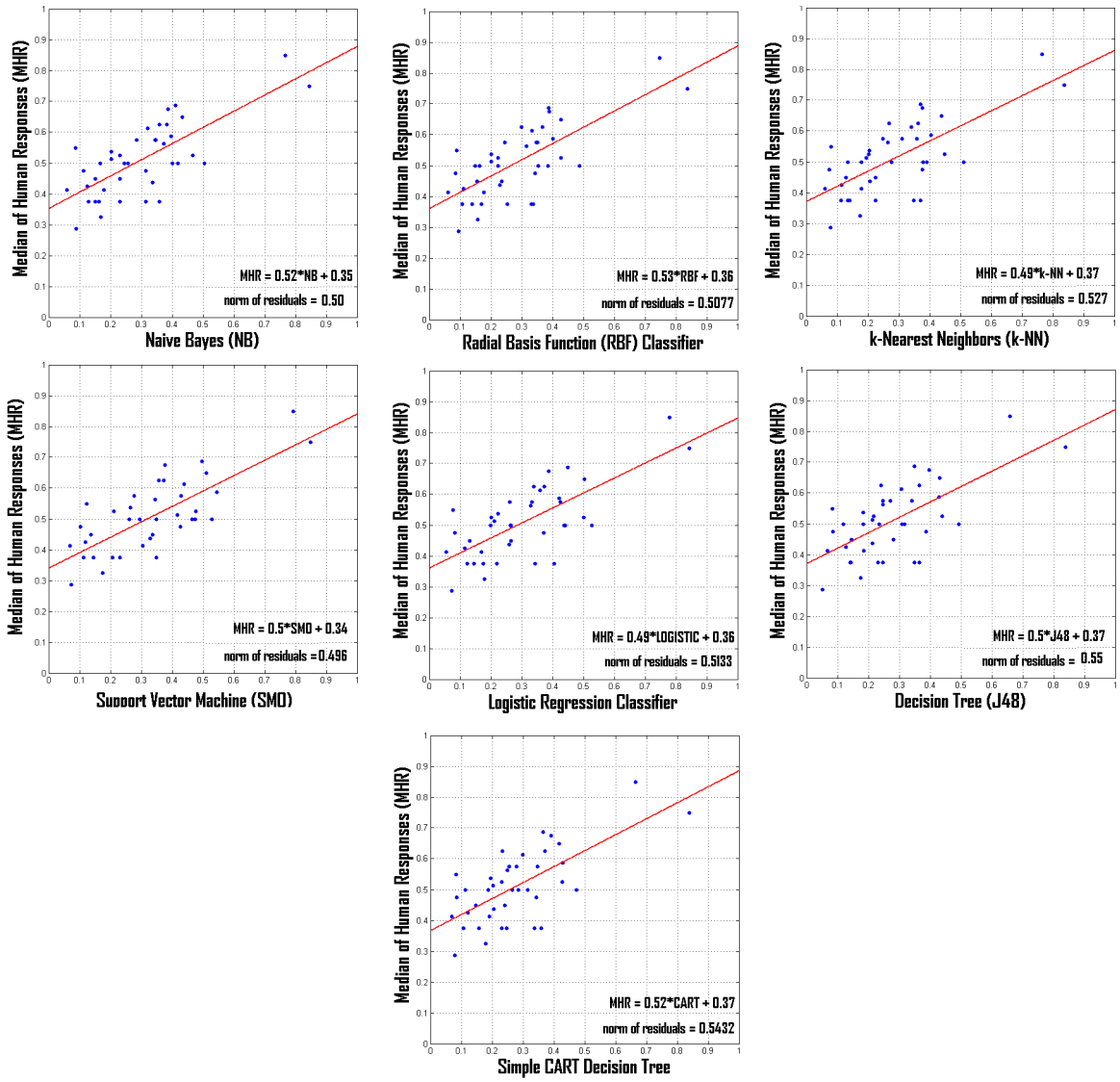


Figure 3.7: Human responses versus the classifier based automated measures on all scatterplots

not surprising that performance of the classifiers will be correlated to human perception in judging the separation between groups. Classification algorithms were designed to learn how each group differs from each other. There can be a number of explanations as to why the humans and the classifiers differ in their judgement of the quality of the views:

- Classifiers can reach high accuracy (low error rate) as long as there is a margin of separation between the classes regardless of the length of the margin, provided that it is numerically significant. But for humans, the margin needs to be perceptually significant which is most possibly

higher than a classifier would be happy with.

- Classifiers would not have a problem even if members of the same class are scattered into separate regions in the visualization. For example; in the view shown in figure 3.8, members of each group form two distinct elongated clusters which are spatially distant from each other. A classifier does not have a problem in separating members of the groups with high accuracy (for example, the Logistic Regression classifier would rate it $\sim$0.1–very good). But, according to the participants, the median rating was around 0.55–average.

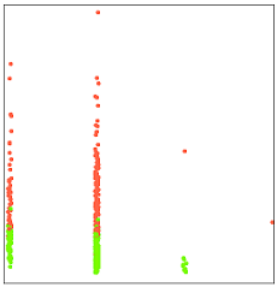


Figure 3.8: Example view from the WDBC dataset

In this regard, explicit measures of compactness versus separation could be useful. Figure 3.9 presents correlation plots for the remaining automated measures including the four measures that explicitly model compactness versus well-separation (LDA index and the three cluster validity indices). The ratings generated by the two clustering validity methods (Davies-Bouldin Index and Dunn Index) have been range-normalized to [0-1] after all images were rated, therefore these values represent ratings relative to the group of scatterplots included in this study.

Tables 3.5- 3.8 present the rankings of the automated measures for each individual dataset. The results vary as we study the outcome with respect to each individual dataset. For the WDBC and Segments datasets, three of the measures (Davies-Bouldin Index, LDA Index and C Index) do not correlate with the participant responses at a statistically significant level. For the Wine dataset, all automated measures are correlated with the human responses and for the Olive oils dataset, all but Dunn Index show statistically significant correlation at $\alpha = 0.05$ level.

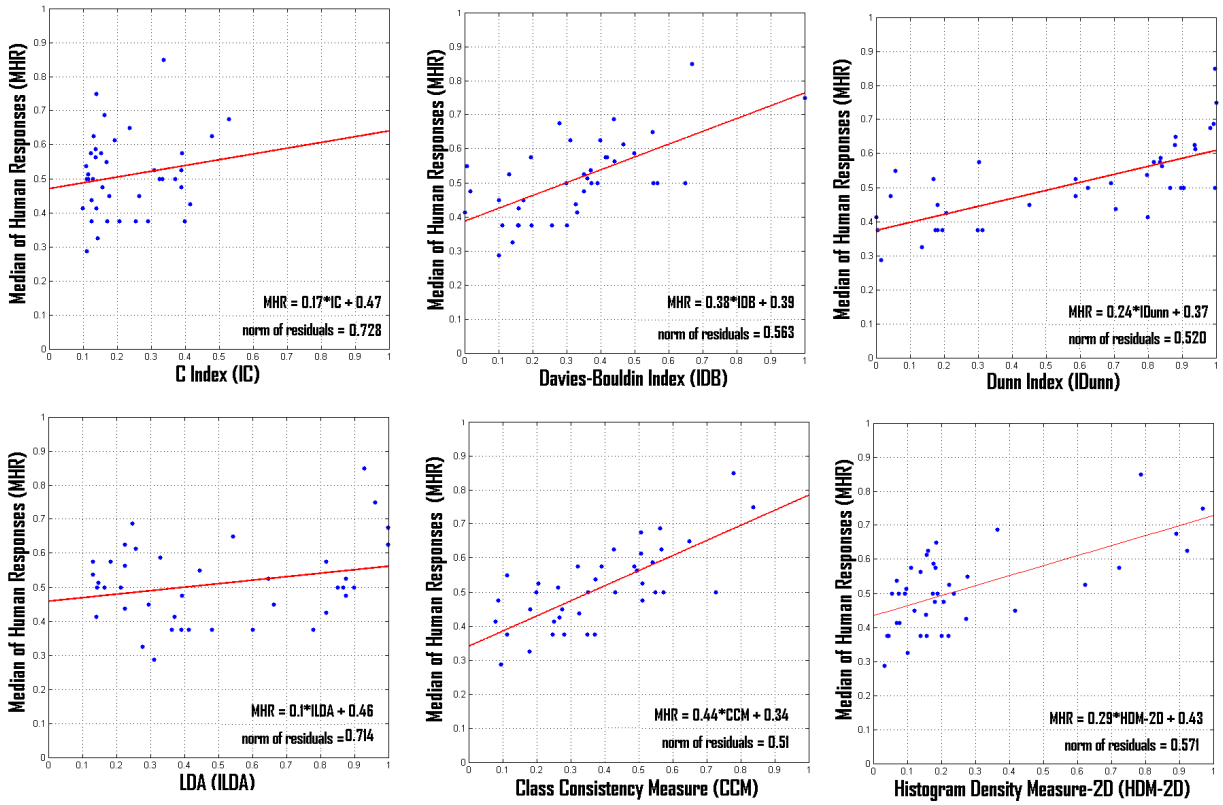


Figure 3.9: Human responses versus clustering validity and other automated measures on all scatterplots

Measure	SSE	R^2	Adj. R^2	RMSE	p-value
2D-Histogram Density Measure	0.0208	0.7592	0.7291	0.0510	< 0.05
Support Vector Machine	0.0216	0.7495	0.7182	0.0520	< 0.05
Radial Basis Function Classifier	0.0272	0.6844	0.6450	0.0583	< 0.05
K-Nearest Neighbors	0.0277	0.6786	0.6385	0.0589	< 0.05
Naive Bayes	0.0280	0.6759	0.6353	0.0591	< 0.05
Logistic Regression Classifier	0.0285	0.6691	0.6278	0.0597	< 0.05
C4.5 Decision Tree	0.0288	0.6667	0.6250	0.0600	< 0.05
Simple Cart Decision Tree	0.0307	0.6436	0.5991	0.0620	< 0.05
Dunn-Index	0.0340	0.6057	0.5564	0.0652	< 0.05
Class Consistency Measure	0.0481	0.4430	0.3733	0.0775	< 0.05
Davies-Bouldin Index	0.0559	0.3516	0.2706	0.0836	0.07
LDA Index	0.0637	0.2616	0.1693	0.0892	0.13
C Index	0.0661	0.2337	0.1379	0.0909	0.16

Table 3.5: Summary of linear relationships (Df:8, $\alpha = 0.05$) between the automated measures and human perception for WDBC dataset (2 classes)

When we look at the results on individual datasets (tables 3.5- 3.8), we notice that the Support Vector Machine is no longer one of the top performers for datasets with larger number of classes (Segments and Olive oil). The Dunn Index shows a significant match for all datasets accept for the Olive oils dataset which might be due to the fact that, the class structures contain outliers and in some cases, the members of the same class are clumped together in multiple areas on the scatterplot. Overall, we

Measure	SSE	R^2	Adj. R^2	RMSE	p-value
Dunn Index	0.0087	0.8061	0.7818	0.0329	< 0.05
Support Vector Machine	0.0118	0.7363	0.7033	0.0384	< 0.05
Class Consistency Measure	0.0145	0.6764	0.6359	0.0426	< 0.05
C4.5 Decision Tree	0.0155	0.6535	0.6102	0.0440	< 0.05
C Index	0.0167	0.6266	0.5799	0.0457	< 0.05
Logistic Regression Classifier	0.0180	0.5988	0.5487	0.0474	< 0.05
Naive Bayes	0.0190	0.5761	0.5231	0.0487	< 0.05
LDA Index	0.0191	0.5741	0.5209	0.0488	< 0.05
Radial Basis Function Classifier	0.0194	0.5675	0.5134	0.0492	< 0.05
Davies-Bouldin Index	0.0201	0.5519	0.4959	0.0501	< 0.05
Simple Cart Decision Tree	0.0205	0.5426	0.4854	0.0506	< 0.05
K-Nearest Neighbors	0.0208	0.5357	0.4777	0.0510	< 0.05
2D-Histogram Density Measure	0.0245	0.4531	0.3847	0.0553	< 0.05

Table 3.6: Summary of linear relationships (Df:8, $\alpha = 0.05$) between the automated measures and human perception for Wine dataset (3 classes)

Measure	SSE	R^2	Adj. R^2	RMSE	p-value
2D-Histogram Density Measure	0.0459	0.7140	0.6783	0.0757	< 0.05
Dunn Index	0.0617	0.6152	0.5671	0.0878	< 0.05
Logistic Regression Classifier	0.0627	0.6092	0.5603	0.0885	< 0.05
Radial Basis Function Classifier	0.0634	0.6049	0.5555	0.0890	< 0.05
Naive Bayes	0.0635	0.6041	0.5546	0.0891	< 0.05
Support Vector Machine	0.0636	0.6035	0.5539	0.0891	< 0.05
K-Nearest Neighbors	0.0708	0.5582	0.5030	0.0941	< 0.05
Simple Cart Decision Tree	0.0712	0.5562	0.5008	0.0943	< 0.05
C4.5 Decision Tree	0.0766	0.5222	0.4625	0.0979	< 0.05
Class Consistency Measure	0.0906	0.4352	0.3647	0.1064	< 0.05
Davies-Bouldin Index	0.1134	0.2927	0.2043	0.1191	0.11
LDA Index	0.1467	0.0847	-0.0297	0.1354	0.41
C Index	0.1603	0.0002	-0.1248	0.1416	0.98

Table 3.7: Summary of linear relationships (Df:8, $\alpha = 0.05$) between the automated measures and human perception for Segment dataset (7 classes)

Measure	SSE	R^2	Adj. R^2	RMSE	p-value
Logistic Regression Classifier	0.0092	0.7802	0.7527	0.0339	< 0.05
Radial Basis Function Classifier	0.0103	0.7539	0.7231	0.0359	< 0.05
Naive Bayes	0.0108	0.7420	0.7097	0.0368	< 0.05
Davies-Bouldin Index	0.0109	0.7405	0.7080	0.0369	< 0.05
K-Nearest Neighbors	0.0114	0.7275	0.6935	0.0378	< 0.05
Class Consistency Measure	0.0122	0.7100	0.6738	0.0390	< 0.05
C Index	0.0138	0.6702	0.6289	0.0416	< 0.05
2D-Histogram Density Measure	0.0147	0.6496	0.6058	0.0428	< 0.05
C4.5 Decision Tree	0.0149	0.6445	0.6001	0.0431	< 0.05
Simple Cart Decision Tree	0.0165	0.6054	0.556	0.0455	< 0.05
Support Vector Machine	0.0188	0.5513	0.4952	0.0485	< 0.05
LDA Index	0.0212	0.4950	0.4319	0.0514	< 0.05
Dunn Index	0.0367	0.1252	0.1252	0.0677	0.32

Table 3.8: Summary of linear relationships (Df:8, $\alpha = 0.05$) between the automated measures and human perception for Olive Oils dataset (9 classes)

found that Davies-Bouldin Index, Dunn Index, C Index and LDA Index might not correlate with human perception, depending on the dataset while all others seem to correlate to some extent. Amongst the classification algorithms, the decision tree algorithms (J48 and Simple CART) did not rank as high as the others in most cases. Considering the way a decision tree hierarchically partitions the space strictly parallel to the visualization axes (such as in figure 2.6), it does not quite match how a human would partition the same view.

From these results, we infer that the degree of match between the human and the automated mea-

sure might depend on the characteristics of the views such as the shape of the clusters formed by the class members. Therefore, we hypothesize that a combination of these measures might model the human perception better than any single measure. In the next section, we derive a composite measure based on the individual measures and investigate its performance across all individual datasets included in our experiments.

3.2.6 Combining the Automated Visual Interpretability Measures

Given the automated measures and the median of human responses for all datasets, we cast a prediction problem that would learn a linear model for the human responses in terms of the automated measures. We trained linear regression models with leave-one-out cross-validation. The following three[2] alternative linear combinations of the automated measures were found using Weka [4].

Model	Expression
M_1	$0.5627 * CCM + -0.4867 * I_C + -0.387 * I_{DB} + 0.0243 * I_{DUNN} + -0.2251 * J48 + -0.2199 *$ K-NN$+$ $0.0094 * I_{LDA} + -0.7833 *$ NAIVE BAYES $+ 0.6506 *$ SMO $+ 0.3874 *$ HDM-2D $+ -0.8589 *$ CART$+$ $0.6282 *$ LOGISTIC $+ 0.7374 * RBF + 0.387$
M_2	$0.6408 * CCM + -0.5197 * I_C + -0.3853 * I_{DB} + -0.7306 *$ NAIVE BAYES $+ 0.6731 *$ SMO $+ 0.4094 *$ HDM-2D$+$ $-1.1535 *$ CART $+ 0.5152 *$ LOGISTIC $+ 0.5605 *$ RBF $+ 0.3948$
M_3	$0.6928 * CCM + -0.4885 * I_C + -0.4515 * I_{DB} + 0.7807 *$ SMO $+ 0.3624 *$ HDM-2D $+ -0.8179 *$ CART $+ 0.3709$

Table 3.9: Three linear models generated using Linear Regression algorithm

Table 3.10 summarizes the degree of match between each combination of automated measures and the human responses across all views. All combined measures rank higher than any single measure on all scatterplots reported on table 3.4.

Amongst these measures, we chose M_2 as the best tradeoff between simplicity (number of measures combined) and degree of match (maximum R^2) and further explored its performance on individual datasets.

Combined Measure	# measures combined	SSE	R^2	Adj. R^2	RMSE	Df	p-value
M_1	13	0.0662	0.8791	0.8759	0.0417	38	< 0.05
M_2	9	0.0672	0.8772	0.8740	0.0420	38	< 0.05
M_3	6	0.0753	0.8625	0.8589	0.0445	38	< 0.05

Table 3.10: Comparison of the three alternative combined measures ($\alpha = 0.05$)

[2]These three models were found using the Linear Regression with feature selection option in Weka. The choices were: no feature selection, Greedy method and the M5 method.

The combined measure M_2 has the highest ranking for WDBC and Segment datasets. For Olive oils data, it ranks 2nd and for the Wine dataset, it ranks 7th (in the middle).

Dataset	SSE	R^2	Adj. R^2	RMSE	Df	p-value
WDBC	0.0097	0.8880	0.8740	0.0348	8	< 0.05
Wine	0.0181	0.5956	0.5451	0.0476	8	< 0.05
Segment	0.0120	0.9246	0.9152	0.0389	8	< 0.05
Olive oils	0.0095	0.7730	0.7446	0.0345	8	< 0.05

Table 3.11: The degree of match between the combined linear model (M_2) of the automated measures and human perception on dividual datasets ($\alpha = 0.05$)

As it is shown in figure 3.10, all three aggregate measures almost perfectly match the ratings given by the participants. This is the natural consequence of training a linear regression model based on the automated measures. The resulting measures are more like *a little bit of this measure and a little bit of that measure* matching the human ratings numerically. The expressions are neither intuitive nor easily interpretable. They are merely the best linear combinations of a subset of the automated measures.

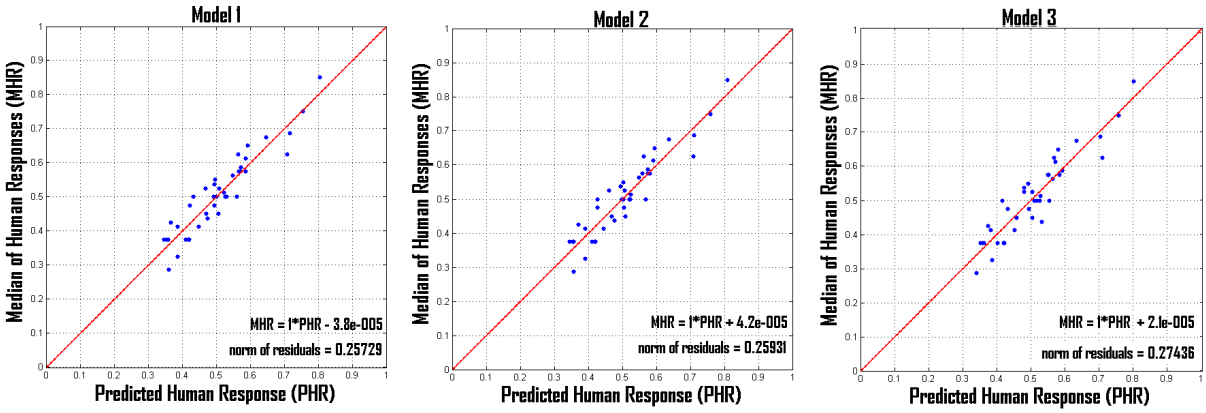


Figure 3.10: Human responses versus the inferred composite models of automated measures on all scatterplots

Also, training a linear model on the set of all scatterplots and applying it to the individual datasets showed that the combined model is not guaranteed to be the most correlated measure to the human perception on each separate case. Moreover, the more the number of measures that are aggregated, the higher the computational cost will be. Even though the four measures that aim to capture compactness versus well-separation (the LDA index and the three cluster validity indices) did not work well for those views where the clusters were not compact, they are still valid options if the task is to detect

compact and well-separated visualizations.

If the data analysis task aims to find 2D visualizations of data as a dimensionality reduction method for prediction, then a classification algorithm would be a viable option. In this case, the visualization/-classifier pair with the best estimated performance will be the choice. The quality of the visualization can also be assessed simultaneously by an additional measure such as any of the cluster validity indices to ensure compactness and separability. Such a multi-objective search algorithm is the central idea developed throughout this thesis.

3.3 User Study on Interpretability of Data Transformation Functions

The goal of the semantic interpretability study is to understand how easily the users interpret and understand mathematical expressions of variables, coefficients and operators that would make up a linear or non-linear projection (transformation) function characterizing the relationship between a set of variables and result of the projection. We developed a software application that automatically administered the experiment and recorded participant responses without investigator intervention. The participants used a consumer grade tablet pen interface connected to the computer via a USB connection.

3.3.1 Participants

The same 20 participants (section 3.2.1) that were involved in the visualization study also took part in this experiment. Before starting the study, all participants were given time to train on using the tablet pen interface.

3.3.2 The expressions

We created 30 mathematical expressions consisting of five possible variables t, u, x, y, z, numerical coefficients, mathematical operators $+, -, *, /$, logarithm, square-root, exponential and power. Undisclosed to the participants, the first five expressions were used as *calibration expressions*.

Expression	# Operands	# Operators	Tree Depth	# Blocks	Total Size
x^2	1	1	2	1	2
$x * y + z * u$	4	3	3	2	7
$2 * log(z) + \sqrt{x}$	3	4	4	2	7
$e^{\sqrt{t+u}\ /\ (\ log(x)\ *\ (\ log(u)+log(z)\ )\ )}$	5	9	6	3	14
$(\ 0.5 * t * \sqrt{(u * y) + t}\ )\ /\ (e^{\sqrt{x * log(y)}\ /\ z})$	8	11	7	3	19

Table 3.12: Five mathematical expressions used as calibration stimuli

The purpose of these initial expressions was to establish a range of complexity of the expressions that would be shown further on. The participant responses to these expressions were not included in analysis of the results. In our experiments, the size of the shortest expression was 2 and the longest expression was 19.

Expression	# Operands	# Operators	Tree Depth	# Blocks	Avg. Block Size	Total Size	Median of Human Ratings	Median of Time Spent Writing (seconds)	# Correct (out of 20) P(Correct)
$log(x)$	1	1	2	1	2	2	0.0	18.69	20 (1.0)
$0.5 * t$	2	1	2	1	3	3	0.025	17.55	20 (1.0)
$x\ /\ (y - 1)$	3	2	3	2	2	5	0.125	19.6	18 (0.9)
$\sqrt{log(y)}\ /\ y$	2	3	4	2	2	5	0.3	21.1	16 (0.8)
$log(t) + log(u)$	2	3	2	2	2	5	0.225	21.15	20 (1.0)
$e^{\sqrt{log(t^2)}}$	1	4	5	1	5	5	0.25	20.37	19 (0.95)
$u * (z + x)$	3	2	3	2	2	5	0.2	19.08	19 (0.95)
$t + (x * z)$	3	2	3	2	2	5	0.25	23.19	19 (0.95)
$0.2 * t + u * y$	4	3	3	2	3	7	0.25	23.48	20 (1.0)
$e^{(t+u)\ *\ (z-1)}$	4	4	4	2	3	8	0.475	17.46	13 (0.65)
$(t + u + x)\ *\ (y - z)$	5	4	4	2	4	9	0.375	20.61	13 (0.65)
$(\ (x * y) - z\ )\ /\ (t - u)$	5	4	4	2	4	9	0.5	24.24	11 (0.55)
$(\sqrt{t} + (z * x) + \sqrt{u})^2$	4	6	5	3	2.33	10	0.56	17.47	11 (0.55)
$y\ /\ (\ (\ (t^2 + u) * z\ ) + x\ )$	5	5	6	3	2.67	10	0.612	19.83	6 (0.3)
$t\ /\ (\ (\ (t^2 - t) * u\ ) - u\ )$	5	5	6	3	2.67	10	0.625	22.45	10 (0.5)
$(\ (t * x)/u\ )\ -\ (\ (y - z)/x\ )$	6	5	4	2	5	11	0.625	17.44	2 (0.1)
$log(x) + log(y) - log(t) - log(x)$	4	7	5	4	2	11	0.375	15.39	12 (0.6)
$(z/x)\ +\ (\ (x/y)\ /\ (t/u)\ )$	6	5	4	3	3	11	0.625	26.05	7 (0.35)
$e^{(\sqrt{t}+\sqrt{u})\ *\ (x+(y*z))}$	5	7	5	2	5	12	0.712	21.85	7 (0.35)
$log(\ (\sqrt{t} * \sqrt{u})\ +\ (x * (t + u)\ )\ )$	5	7	5	2	5	12	0.75	22.64	6 (0.3)
$((x - 1) * (y - 2))\ /\ (t/u)^2$	6	6	4	3	3.33	12	0.575	19.07	10 (0.5)
$(\ (log(t) * \sqrt{u}) + (t + x) - y^2\ )\ /\ (t * z)$	7	9	5	3	4.67	16	0.85	18.41	0 (0.0)
$(\ (0.2 * e^{-2*x})\ /\ (z + x)\ ) - (\sqrt{y}/(u * x)\ )$	8	9	6	3	5	17	0.875	19.27	0 (0.0)
$(t * x) + (u * y) + (z * y) + (x * u) + (t * z)$	10	9	5	5	3	19	0.637	19.63	2 (0.1)
$0.1 * t + 0.5 * u + 0.2 * z + 0.4 * y + 0.3 * x$	10	9	5	5	3	19	0.75	22.16	1 (0.05)

Table 3.13: 25 mathematical expressions used in assessment of expression interpretability (ordered by total size)

3.3.3 Task

The participants were informed that they would be shown a series of mathematical expressions of five possible variables t, u, x, y, z, numerical coefficients, mathematical operators $+, -, *, /$, logarithm, square-root, exponential and power. They were told that each expression would be displayed for 10 seconds and that their task was to study the expression within that time and write it back using the tablet pen after the expression was removed from the screen.

The participants were also asked to rate how easy it was to understand/interpret the given expression. The rating was on a continuous scale from 0 (very easy) to 1 (very difficult) with labels shown on table 3.14.

Rating	Value
Very Easy	0.0
Easy	0.25
Average	0.5
Difficult	0.75
Very Difficult	1.0

Table 3.14: Labels for semantic interpretability ratings on continuous scale

3.3.4 Methodology

After the first five calibration expressions, the remaining 25 expressions (table 3.13) were then shown in a randomized order to each participant. The expressions were presented to the participants in a linearized form. Specifically, we chose not to display the division operation as a fraction in order not to create a visual cue that would make it easier to interpret the expression as opposed to addition, subtraction or multiplication.

For each participant, we recorded the time they spent writing each expression and their rating on how easy it was to understand the expression. The images of the expressions they wrote down were automatically captured and saved to disk for manual inspection for correctness (figure 3.12). For this study, we only considered correct/incorrect response rather than assessing partial correctness.

Figure 3.11: Screenshots from the user study on interpretability of transformation functions

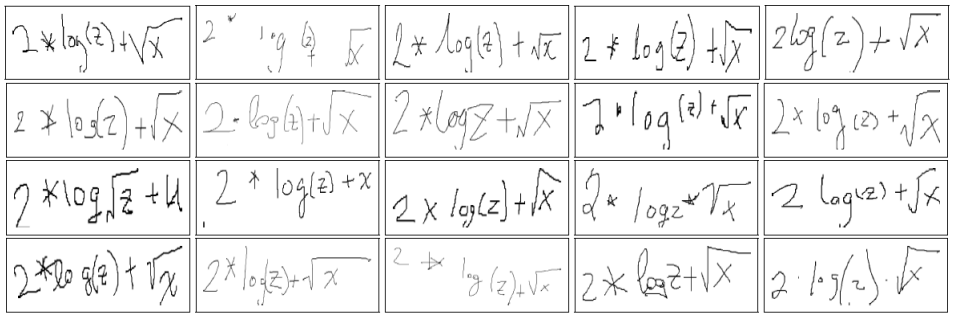


Figure 3.12: A sample of participants' entries using tablet pen interface (expression: $2 * log(z) + \sqrt{x}$)

3.3.5 Results

The outcome of the study is summarized on table 3.13. For each expression, we report the median value of the participant ratings, median value of the total time it took for the participants to write down and rate the expression and the number of correct responses. We first examined the relationship between how an expression is rated by the participants and how frequently it was written down correctly. We hypothesized that the expressions that were frequently replicated incorrectly would also be rated as difficult to interpret by the participants. Indeed, we found that there was a strong correlation between them (Pearson's R=0.9379, Df=23, $p < 0.05$) indicating that the ratings given by the participants were consistent with their observed behavior (answering correctly/incorrectly). We did not find a meaningful relationship between the time taken to write the expression and the subjective rating. Most possibly, it is due to the fact that the participants did not spend much time on those expressions that they found hard to interpret. Therefore, we will present only the results based on the participant ratings here.

In order to define a relationship between the structure of the expression and the participant ratings, we determined various attributes of the expressions characterizing the complexity. For this purpose, we utilized a syntax-tree representation for the expressions (figure 3.13).

The number of operands include the variables and other numerical values. A block is a sub-expression composed of multiple operators and operands. The tree depth relates to the nestedness in the expression and the number and size of the blocks indicate the distinguishable components it contains.

Table 3.13 shows the values of the six derived attributes and the participant ratings for each expres-

sion. The results reveal that up to expression size 7 (up to 4 operands or 4 operators), the participants rated the expressions to be very easy-easy (rating ≤ 0.3).

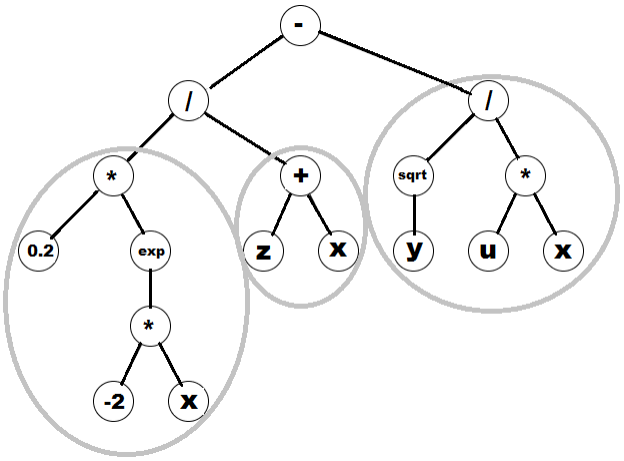


Figure 3.13: Expression tree for $(\ (0.2 * e^{-2*x})\ /\ (z + x)\ ) - (\sqrt{y}/(u * x)\ )$. Tree depth:6, # of blocks:3, #operands:8, #operators:9

We first looked at how much each of these attributes can predict the participant's ratings on interpretability (table 3.15). Unsurprisingly, the number of operators and total size affect how the expressions are rated. But this does not explain why the expressions of the same size were rated differently by the participants.

Expression Attribute	SSE	R^2	Adj. R^2	RMSE	p-value
Number of Operators	0.3015	0.8023	0.7937	0.1145	< 0.05
Total Size	0.3130	0.7948	0.7859	0.1167	< 0.05
Number of Operands	0.5177	0.6606	0.6458	0.1500	< 0.05
Tree Depth	0.5290	0.6532	0.6381	0.1517	< 0.05
Number of Blocks	0.9704	0.3638	0.3361	0.2054	< 0.05
Avg. Block Size	1.0060	0.3406	0.3119	0.2091	< 0.05

Table 3.15: Summary of linear relationships (Df:23, $\alpha = 0.05$) between the expression attributes and human ratings on complexity

We cast a regression problem that learns a mapping between the structural properties of an expression and the degree of human interpretability as reported by our participants. We trained a linear model with leave-one-out cross-validation.

$$
\begin{aligned}
PredictedHumanRating(PHR) = \\
& 0.0854 * \text{Tree Depth} + \\
& -0.2568 * \text{Number of Blocks} + \\
& -0.1014 * \text{Avg. Block Size} + \\
& 0.0899 * \text{Total Size} + \\
& 0.2151
\end{aligned}
$$

This linear model is highly predictive of the human ratings with respect to tree depth, number and average size of the blocks, and the total size (Pearson's R=0.9598, Df=23, $p < 0.05$). Based on this model, we infer that humans rate longer and nested expressions as more difficult to interpret while the existence of small number of compact blocks increase the interpretability.

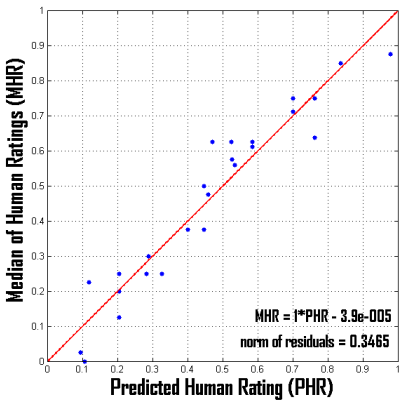


Figure 3.14: Human ratings versus the predicted ratings for the expressions

3.4 Discussion

We investigated the relationships between human perception and automated measures that aim to quantify interpretability. For visual exploration of high dimensional labeled datasets, we considered two forms of interpretability. Visual interpretability is concerned with how easy it is to tell the members of different classes apart by looking at 2D scatterplots of data. We argued that classifier performance and various clustering validity measures from the machine learning literature can also be used to assess the quality of the views besides the recently proposed visual quality measures. We presented a user study on four datasets with varying number of classes comparing a large number of automated measures to human perception. Our results indicated that no single measure outperforms others on all datasets. While the Davies-Bouldin, Dunn Index, C Index and LDA Index might not correlate with human perception well for all datasets, they work fine if the visualization is composed of compact and relatively well separated clusters. All other measures correlate with human ratings at a statistically significant level across all datasets.

A linear combination of a subset of the automated measures trained on the set of all scatterplots correlated significantly well with the human perception on all individual datasets, although it was not always the highest ranking measure. However, the linear expressions combining the measures are neither intuitive not interpretable. Moreover, the computational burden increases as more measures are combined.

The implication of the results of our user study on visual interpretability is that, any of the automated measures we studied can be used to judge the separability of the class structures on scatterplots. In an exploratory data analysis setting, a classification algorithm can be used to search for the best lower dimensional data representation/classifier pair for predictive analysis. A separate measure of compactness and separability can also be used simultaneously to reinforce the ease of interpretability of the visualization. Such a multi-objective search algorithm is the central idea developed throughout this thesis.

Semantic interpretability is concerned with how easy it is for humans to understand the data transformation functions that project the original features into lower dimensions. In our study, we focused on *readability* of the expressions as an indicator of interpretability. We investigated how humans would rate expressions of varying levels of structural complexity. We found that up to expression size 7 (up to 4 operands or 4 operators), the participants rated the expressions to be very easy-easy (rating $\leq$ 0.3). Based on a linear combination of various structural properties of an expression, we inferred that humans rated longer and nested expressions as more difficult to interpret while the existence of small number of compact blocks increase the interpretability.

In exploratory analysis of labeled data, simple feature-feature combinations might not always be the best views that reveal the class structures. Linear or non-linear feature transformation functions might create better 2D views. However, the space of possible transformation functions is vast. Therefore, automated complexity measures reflecting the human perception closely will be useful in finding interpretable data transformations.

The automated measures for visual and semantic interpretability of data transformations can be com-

bined in a number of ways in order to search for good views of data that are also easily understandable in terms of the original attributes. A weighted linear combination of visual and semantic interpretability measures can be utilized or they can be optimized simultaneously using a multi-objective optimization scheme.

Chapter 4

On Comparability and Reproducibility of Data Classification Results

Evolutionary computing methods are attracting more researchers who are working on data mining problems. As in all research fields, it is important that the reported results are comparable and reproducible. The motivation for the study presented in this section came from an anonymous reviewer comment on the statistical test that we had used to report the performance of our results for an earlier paper. When we looked at the papers employing similar techniques (using genetic programming for feature extraction or classification), we realized that there was no established standard within the community. Then, we turned to the machine learning literature to see if there were any established rules of thumb or best practices across the field. Since our work is about data classification, we focused on papers addressing this problem. Not surprisingly, there is no universally accepted way of conducting and reporting experiment results on classification problems. On the other hand, we found that there were a number of major issues raised by the researchers scattered into various publications. In this section, we review these issues and present a standardized way that we followed in order to ensure consistency across our experiments.

There are multiple aspects of a data classification task and for each aspect there are certain choices to be made. Figure 4.1 summarizes the questions centered around a typical classification problem.

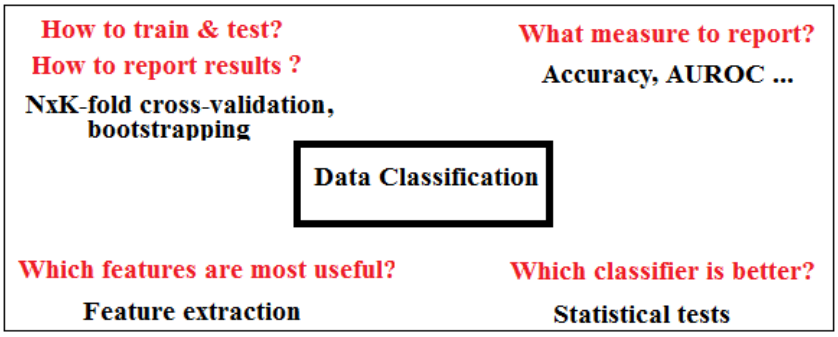


Figure 4.1: Questions related to a data classification task

It has been a common practice in machine learning to report the performance of a supervised learning algorithm in terms of an *average* measure of performance (such as classification accuracy, area under ROC, f-measure [49, 47]) based on multiple random subsets of the dataset. The goal is to compute a fair measure of how the learning algorithm would generalize to unseen data. Keeping a separate test set to evaluate the learning algorithm helps us ensure that the algorithm is actually learning the underlying regularities in the problem rather than the noise in the training set which is called *over-fitting*. Over-fitting is simply the case where the algorithm is tuned to the training samples to a point where it just memorizes the training data but does not perform well on a separate sample from the same problem domain.

The most widely used training and evaluation technique is called *cross-validation* which divides the dataset into a number of folds and uses one fold as the test set at a time. Various forms of the general NxK-fold (N times K-fold) cross-validation that are widely used in machine learning are: 1x10 (1 time 10-fold), 5x2 (5 times 2-fold) and 2x5 (2 times 5-fold) which are referred as the *canonical* cross-validation techniques [121]. The total number of runs for these cross-validation methods is 10. A 10x10-fold variation which requires 100 runs is also used. A special case where K is equal to the number of records in the dataset is called leave-one-out cross-validation since only one data point is used as the test set for each run.

Figure 4.2 shows the 1xK-fold cross-validation process. First, the dataset is divided into K-equal sized partitions. Generally, the proportion of class members within each partition is kept roughly the same as in the whole dataset. This technique is called *stratified cross-validation*. After partitioning, the learning algorithm is run K-times by keeping each partition aside as the test set and training the classifier on the remaining partitions. For each run, the classifier performance on each test set is recorded giving a total of K estimates of the classifier's generalization power on the problem under study. Then, the mean and standard variation of these estimates are reported.

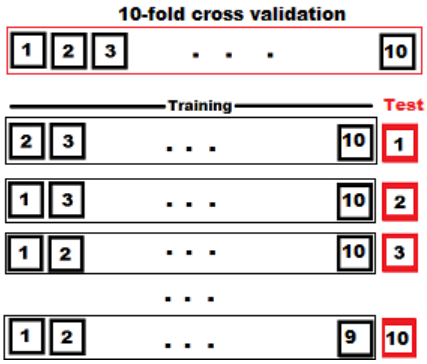


Figure 4.2: One iteration of K-fold cross-validation

The process described above can be repeated for more iterations. For each iteration, the order of the records in the dataset is first randomized and then the K-fold cross-validation is performed. In this scheme, the NxK-fold cross-validation method produces NxK estimates of the classifier's generalization power.

The cross-validation results on the same problem domain can be used to compare different learning algorithms. Assuming that each classifier's generalization power is a random variable represented by a sample of the NxK results computed by the cross-validation process, a statistical test is utilized in order to validate the difference between the algorithms.

In data mining problems, the dataset is collected by a researcher who knows the problem domain hoping that the dataset is actually capturing the essence of the problem. More often than not, the features are just added in abundance since it is not possible to know the true set of underlying features explaining why members of each class are different than members of other classes. Finding out the

most discriminative set of features is part of the machine learning process and it is tightly connected to classification algorithms. Indeed, it has been specified that the boundary between feature extraction and classification tasks was vague. If the feature extraction algorithm is good, then a trivial classifier would work and if the classifier would perform feature extraction, then an explicit feature extraction step might not be necessary (see [49], page 129). For example, the hidden nodes of a multilayer perceptron algorithm encode non-linear combinations of the input as the extracted features and the decision tree algorithm performs feature selection by eliminating the inputs that are not discriminative. The general approach is to perform explicit feature extraction. However, the practice varies. Some researchers apply feature extraction as a pre-processing step on the whole dataset and then run the classification algorithm while others couple the feature extraction step with a classifier.

We identified the following three issues related to performing cross-validation in data classification experiments:

- There is no standard in determining how many folds (K) and repetitions (N) should be used in partitioning the dataset. However, due to the computational burden, most cross-validation experiments (1x10-fold, 2x5-fold or 5x2-fold) include 10 runs. Lower number of folds result in less data for training while higher number of folds creates overlap between the training sets and introduces dependency. Recently, it has been pointed out by Raeder et al. in [121] that the results of algorithm comparisons are extremely sensitive to the ordering of the data, especially for a small number of cross-validation iterations. This indicates a potential issue with the common cross-validation practice since the number of iterations are generally low (at most 10). The randomization of the data during the cross-validation is arbitrary and most researchers do not specify the details of how they performed the randomization. Therefore, most reported algorithm comparisons might not be reproducible thus reducing their credibility.

- It has been pointed out in a number of papers that parametric tests of statistical significance on the cross-validation results may be misleading in the case that the assumptions of these tests

were violated. Therefore, claims of superiority of a learning algorithm based on an unjustified parametric test might not be valid.

- Feature extraction is an integral part of data classification. There is a potential danger of overfitting that is introduced by using an explicit feature extraction (dimensionality reduction) algorithm as a pre-processing step before evaluating the learning algorithm using a cross-validation scheme. The recommended practice is to move the feature extraction into the cross-validation step and couple the feature extraction with classification.

The following sections discuss each above-mentioned aspect of using cross-validation in data classification research and then lay out the experiment process we utilized for the work reported in this thesis.

4.1 The effects of Random Partitioning

The heart of the cross-validation technique is the creation of random partitions of the datasets that are used to evaluate the machine learning algorithms. Bouckaert and Frank [28] and Raeder et al. [121] study how much the random partitioning would affect the comparisons of learning algorithms. Bouckaert and Frank report that 10 iterations of 10-fold cross-validation increase the reproducibility of the results compared to 5x2-fold and 1x10-fold cross-validation based on experiments on a large number of benchmark datasets using the Naive Bayes, Nearest Neighbor and the C4.5 decision tree algorithms.

Raeder et al. show that the conclusions one could draw from arbitrarily generated random partitions would be highly unstable, especially for a small number of iterations. The authors create 500 iterations of 2-fold, 5-fold and 10-fold cross validation partitions using consistent random number seeds, in order to ensure that each experiment would be run on the exact same partitions. They utilize 6 classifiers (2 decision trees, Naive Bayes, Support Vector Machine (SMO), MLP Neural Network and Nearest Neighbors) on a variety of benchmark datasets. They plot the running average test set area under ROC (AUROC) measure for each iteration. The graphs reveal that the relative superiority between the

classifiers fluctuate at the beginning and stabilize at later iterations. Their results indicate that depending on the dataset, more than 10 cross-validation iterations might be needed in order to eliminate the effect of random partitioning.

We have repeated the same experiments and confirmed this finding on a number of datasets. Figure 4.3 shows the results for 50 iterations of 10-fold cross validation on 10 datasets using 7 classifiers. At each iteration, we plot the cumulative average test set weighted AUROC measure for each classifier. For example; at iteration 10, we show the average performance for all iterations up to 10. The values at each iteration serve as a snapshot of classifier rankings up until that many cross-validation iterations. The figures show that for some datasets, the rankings change rapidly for small number of iterations and stabilize as the number of cross-validation iterations increases. We note that the same observation applies regardless of the classifier performance measure.

This finding has two immediate implications. First, the canonical cross-validation techniques (such as 1 time 10-foldcv, 5-times 2-foldcv, 10-times 10-foldcv) are unstable since the number of iterations for the classifier comparisons to stabilize is generally much higher. Therefore, more iterations are needed until a state-steady is reached. However this is highly domain dependent and can be computationally infeasible. Second, most publications utilize a canonical cross-validation technique without a standard randomization scheme. Majority of the UCI benchmark datasets that are being used in publications are rather small and computational power has increased over the years. Therefore, using a canonical cross-validation technique seems to be motivated by the tradition in the community rather than a computational limitation. However, the lack of a consistent way of creating the folds across the research community makes it difficult to compare results on same benchmark datasets. Each researcher creates a separate set of random partitions using different random number seeds. Unless enough number of cross-validation iterations were run to ensure steady state performance, the differences in data randomization effects the outcome of the experiments. Also, the number of folds (K) might affect how fast the estimate of a classifier's performance stabilizes.

Figure 4.5 shows a series of cross-validation runs on the BUPA dataset using two decision tree classi-

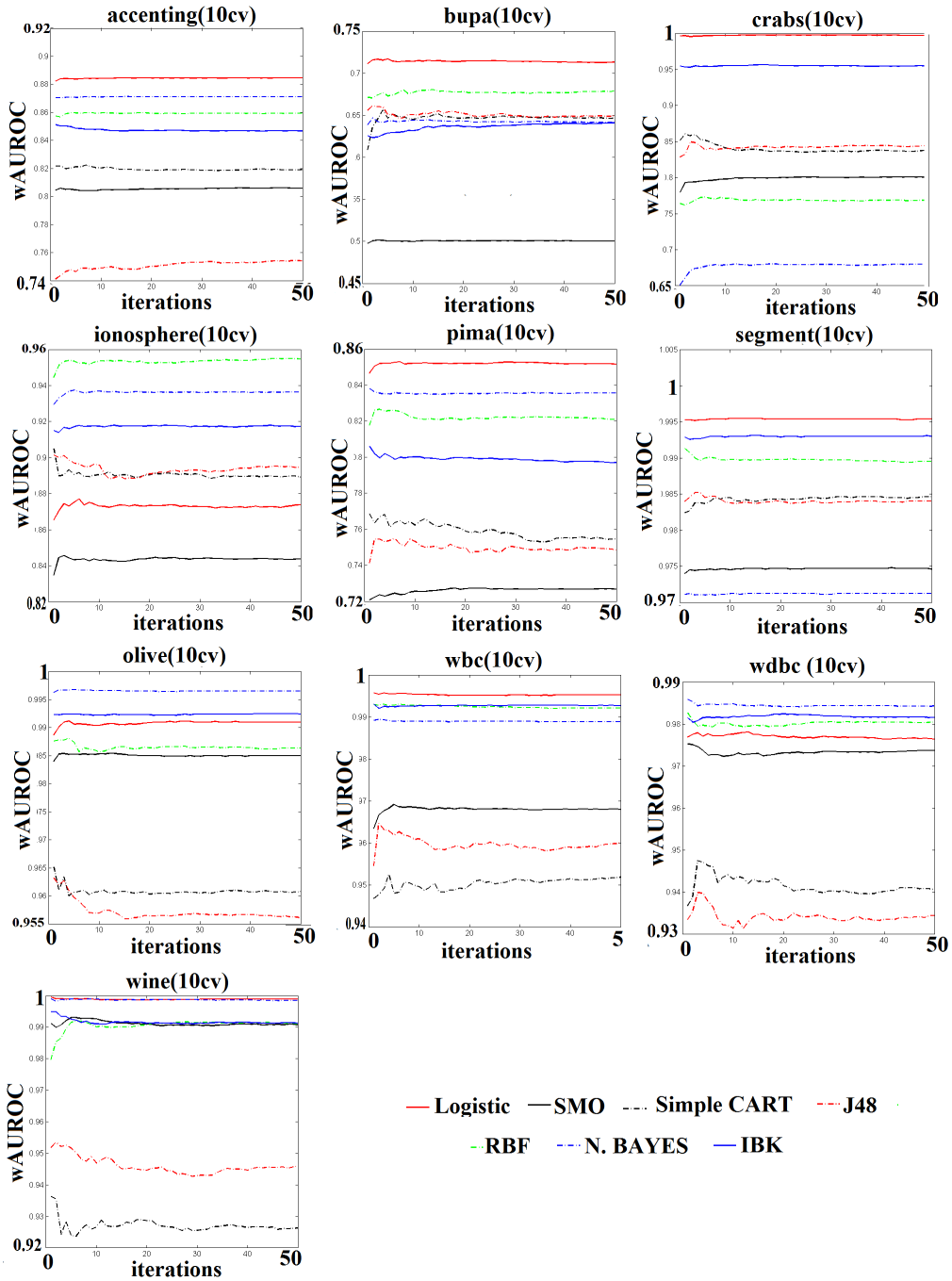


Figure 4.3: Results of up to 50-iterations of 10-fold cross-validation on 10 datasets. Larger wAUROC indicates better performance. Each dataset was randomized using 50 predetermined random number seeds, one for each iteration of the cross-validation process in order to compare each classifier on the same set of test sets.

fiers (Simple CART and J48). These two classifiers perform very similarly on this dataset. We randomly created 50 different sets of 500xK-fold cross-validation partitions. As the plots show, especially for a

small number of iterations, it is not possible to clearly tell which classifier performs better. For 2-fold and 5-fold cross-validation runs, the performance comparison starts to stabilize as the number of iterations goes to 500. This means that, one has to run a 500x2-fold or 500x5-fold cross-validation before it appears that J48 performs better than the simple CART. In the case of 10-fold cross-validation, the performance difference becomes evident as early as 50 iterations and at 500 iterations the difference is very clear.

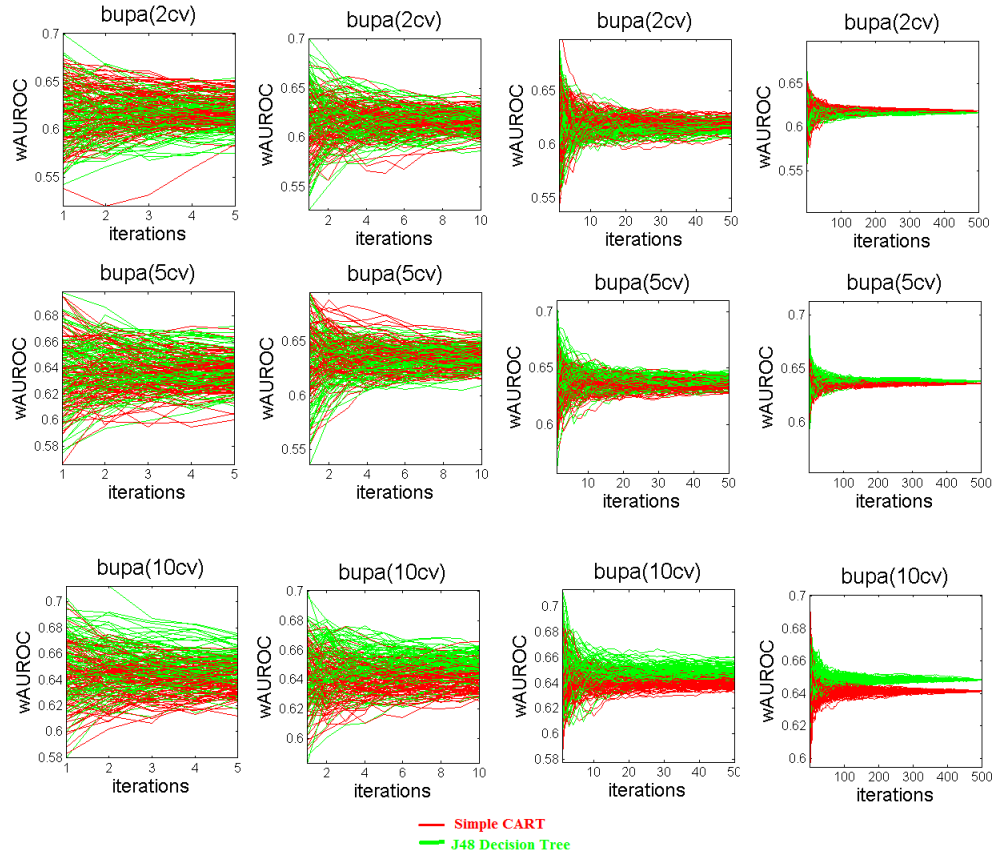


Figure 4.4: Comparison of the J48 and Simple CART classifiers on BUPA dataset using 500xK-fold cross-validation. The difference becomes more clear as more iterations are performed for all K. For K=10, the comparison stabilizes faster on this dataset.

Raeder et al. point out two related but slightly different definitions of consistency in comparing different algorithms based on cross-validation results: reproducibility and replicability. Reproducibility is a wider term which means that the comparison holds regardless of how the randomization was done. As the authors point out, reproducibility can not be guaranteed unless enough number of cross-validation

iterations have been run until a steady state is reached (the performance estimate for each compared algorithm stabilizes). As it is evident from the results presented by the authors and based on our own experiments, the number of cross-validation iterations for each classifier is problem specific and larger than the number of iterations that are used in any canonical method. The authors propose a number of ways to detect when the steady-state is reached as the iterations progress in order to reduce the computational burden.

Replicability is a less strict concept than reproducibility which tells us that under which conditions we can repeat the reported experiments. It does not require for the steady-state point to be reached which lets us compare results using the canonical cross-validation methods. As long as we guarantee that a consistent set of random number seeds are used to generate data for each iteration, we can compare the algorithms at various snapshots as the iterations progress. As for the number of folds (K), the authors recommend a 10-fold scheme. They argue that, on a number of different datasets for which they had a separate test set (true test set) which was not used in the cross-validation scheme, the cross-validation estimates were more consistent with the estimates on the true test set for K=10.

The UCI repository provides a centralized source for datasets. But it does not contain pre-partitioned versions of the data. We believe that such a repository would be very useful to the researchers in reporting and comparing their results. Another option is to establish a standard way of creating the cross-validation partitions. We decided to adopt the scheme (consistent random seeds) used by Raeder et al. in [121] to create cross-validation partitions for our experiments (algorithm 2).

Algorithm 2: Standard random partitioning of datasets for NxK-fold cross validation

```
foreach Cross-Validation Repetition (r=1 .. N) do
    randomSeed = r;
    shuffle(Dataset,randomSeed);
    Stratified_Partitioning(Dataset,K);
    foreach Cross-Validation Fold (f=1 .. K) do
        training_{r,f} = Dataset-Partitiiton_f;
        test_{r,f} = Partitition_f
    end
end
```

The plots we have shown above are based on the average performance over NxK test sets. They demonstrate how the comparison of learning algorithms on the same dataset can be affected by the

random partitioning of the data. In order to make a statistically sound comparison, the whole sample of NxK test set performances (figure 4.5) should be compared instead of just the average which is covered in the next section.

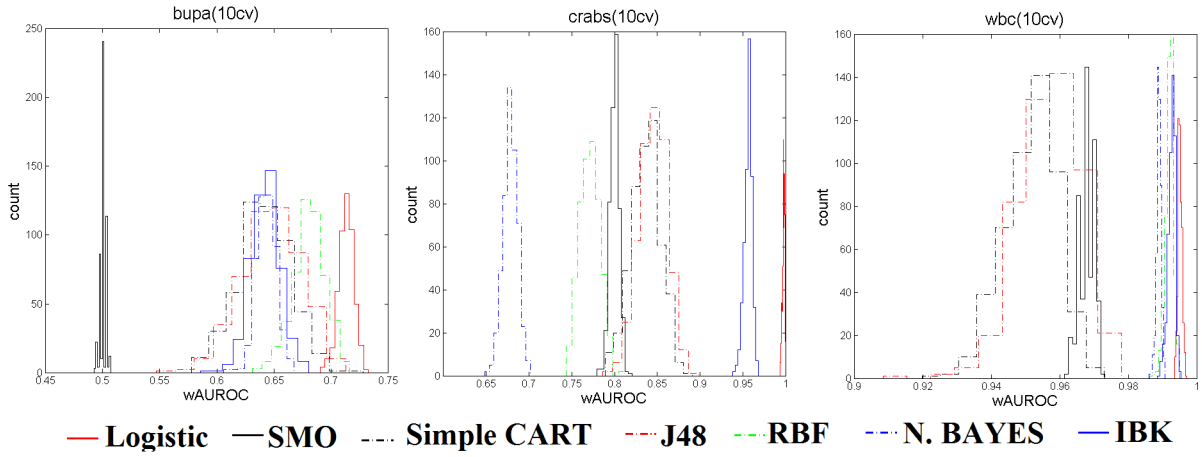


Figure 4.5: Histograms of test set performances over 500x10-fold cross-validation for three datasets

4.2 Statistical Tests for Algorithm Comparisons

A large number of published papers utilize parametric statistical tests in order to report significant difference between two algorithms. Amongst the most common techniques from the machine learning literature are 1x10cv followed by t-test, 5x2cv t-test [45] and 5x2cv F-test [8]. The 5x2cv technique generates non-overlapping training/test set pairs at the expense of using only 50% of the available data for learning. On the other hand, 1x10cv creates highly overlapping training sets thus making the training set classification results correlated. Although there has been an ongoing debate on strength and weaknesses of each algorithm, there is not a standard way of determining how many folds, repetitions and statistical test to use.

Within the context of comparing two learning algorithms on a problem domain (dataset), the goal of a statistical test is to check if both algorithms have the same performance on unseen data. The dataset is divided into a series of training-test sets using a sampling procedure such as cross-validation or bootstrapping (sampling with replacement). A test statistic is defined in terms on performance on

test sets. The null hypothesis is that there is no difference between the algorithms on the same dataset. In a parametric test setting, the defined statistic is assumed to follow a certain distribution (such as t-test, F-test) if the null hypothesis holds. The test computes the probability that the derived test statistic has been drawn from the assumed distribution. If this probability is lower than a threshold, the null hypothesis is rejected indicating that the difference in test set statistic is due to the difference in the algorithms.

The power and Type I error of a statistical test are two related concepts that are used to compare different tests. If the test incorrectly rejects the null hypothesis when actually there is no difference, this is called a Type I error. Conversely, if the test incorrectly accepts the null hypothesis when there is difference, it's called a Type II error. The power of the test is given as $1 - P(\text{Type II error})$. Ideally, a statistical test with lower Type I error and higher power is preferred.

For pair-wise comparison of the learning algorithms paired parametric or non-parametric tests were developed. If more than two algorithms are compared, ANOVA (parametric) and Friedman or Kruskall-Wallis (parametric) tests can be used with the null hypothesis that there is no significant difference amongst the algorithms. Alternatively, a series of of pair-wise comparisons can be made.

4.2.1 Comparing Two Algorithms

Kohavi advocates the use of 10-fold cross validation based on empirical evidence showing that it has lower variance in test set performances than the leave-one-out (LOOCV) or bootstrap estimation (sampling with replacement) methods [82].

A paired t-test following the K-fold (generally 10-fold) cross-validation is performed in order to compare two algorithms as follows:

- Each algorithm is trained on each of the K training sets and tested on the corresponding test sets for a total of K performance estimates (p_k^1, p_k^2).
- For each fold k, the difference $d_k = p_k^1 - p_k^2$ is computed.

- Given that p_k^1 and p_k^2 are (approximately) normally distributed, so is their difference.

- The null hypothesis is that there is no difference between the performance of the two algorithms:

  $H_0 : \mu_{d_k} = 0$

  $H_1 : \mu_{d_k} \neq 0$

  The t-statistic is computed as :

  $t = \frac{\bar{d}-0}{S/\sqrt{K}} \sim t_{K-1}$

  where $\bar{d} = \frac{\sum_{k=1}^{K} d_k}{K}$ and $S^2 = \frac{\sum_{k=1}^{K}(d_k-\bar{d})^2}{K-1}$

- The null hypothesis is accepted at significance level α if $-t_{\alpha/2,K-1} \leq t \leq t_{\alpha/2,K-1}$

According to Diettrich, 10-fold cross validation followed by a paired t-test should not be used since it has higher Type I error due to the fact that training folds are highly overlapping thus causing higher than expected variance in test set performances. Diettrich compares five classical statistical tests and concludes that a *5x2cv paired t-test* has reasonable power and low Type I error [45].

The 5x2cv paired t-test proposed in [45] can be used to compare two algorithms as follows:

- Each algorithm is evaluated on 5 iterations of 2-fold cross-validation scheme for a total of 10 performance estimates.

- For each iteration ($n = 1, .., 5$), the average difference between the two algorithms across all folds (k=1,2) is computed as $\bar{d}_n = \frac{\sum_{k=1}^{K} d_{n,k}}{K}$ where $d_{n,k} = (p_{n,k}^1 - p_{n,k}^2)$.

- Estimated variance for each iteration is computed as $S_n^2 = (d_{n,1} - \bar{d}_n)^2 + (d_{n,2} - \bar{d}_n)^2$

- The null hypothesis is that there is no difference between the performance of these two algorithms. This means that $d_{n,k}$ is (approximately) normally distributed with 0 mean and unknown variance σ^2.

  $H_0 = \mu_{d_{n,k}} = 0$

  $H_1 = \mu_{d_{n,k}} \neq 0$

  Then $d_{n,k}/\sigma$ is approximately unit normal and S_n^2/σ^2 follows a χ^2 distribution with 1 degree of

freedom. Assuming S_n^2s are independent (which is not true since the training/test sets were not independent), the sum is χ^2 with 5 degrees of freedom: $M = \frac{\sum_{n=1}^{5} S_n^2}{\sigma^2} \sim \chi_5^2$ and $t_{n,k} = \frac{d_{n,k}/\sigma}{\sqrt{M/5}} \sim t_5$

- t=$t_{1,1}$ is the selected t-statistic in [45] thus $t = \frac{d_{1,1}}{(\sum_{n=1}^{5} S_n^2)/5} \sim t_5$
- The null hypothesis is accepted at significance level α if $-t_{\alpha/2,5} \leq t \leq t_{\alpha/2,5}$

Aside from the fact that independence between the samples is assumed when it is actually not the case, out of the 10 possible t-statistics, only one of them is considered. Alpaydin points out that the ordering of the iterations would change the outcome of the 5x2cv paired t-test and proposes the *combined 5x2cv paired F-test* that is robust against the ordering of iterations [8]:

- Assuming $d_{n,k}/\sigma$ is unit normal, $d_{n,k}^2/\sigma^2 \sim \chi_1^2$ and the sum R=$\frac{\sum_{n=1}^{5}\sum_{k=1}^{2} d_{n,k}^2}{\sigma^2} \sim \chi_{10}^2$.
- Since R and M are both χ^2 random variables (though not independent) $f = \frac{R/10}{M/5} = \frac{\sum_{n=1}^{5}\sum_{k=1}^{2} d_{n,k}^2}{2\sum_{n=1}^{5} S_n^2} \sim F_{10,5}$.
- The null hypothesis is accepted at significance level α if $f < F_{\alpha,10,5}$.

The 5x2cv paired F-test has been reported to have lower Type I error and higher power than the 5x2cv t-test.

Bouckaert and Frank claim that the concept of replicability of the results in common cross-validation schemes is also important besides the type I error and power of the statistical test [28]. They point out that algorithm comparisons based on only 10 estimates as in the 1x10-fold or 5x2-fold cross-validation schemes lead to results that are not replicable due to effects of random ordering of the cross-validation partitions. Later, this finding has been further confirmed by Raeder et al. in [121]. Bouckaert and Frank proposed a 10x10-fold cross validation scheme in order to increase replaceability of the results. The authors propose a modified version of paired t-test that suffers from high type I error due to the inherent dependence between the training (across folds and iterations) and test sets (across iterations). This modification is based on the variance correction for random subsampling that was proposed by Nadaeu

and Bengio in [108]. The modified test that is called the *corrected repeated k-fold cv test* works as follows:

- For N iterations of K-fold cross-validation, the difference in test set performance of the two algorithms are computed as: $d_{n,k} = p^1_{n,k} - p^2_{n,k}$.
- The mean of the differences is: $\bar{d} = \frac{\sum_{n=1}^{N}\sum_{k=1}^{K} d_{n,k}}{NxK}$
- Sample variance is : $S^2 = \frac{\sum_{n=1}^{N}\sum_{k=1}^{K}(d_{n,k}-\bar{d})^2}{NxK-1}$
- Assuming $d_{n,k}$s are independent,

  $t = \frac{\bar{d}}{S/\sqrt{NxK}} \sim t_{NxK-1}$

  However, the independence assumption is flawed and this test has very high Type I error. It was empirically confirmed that, as N and K increase, the value of the t-statistic increases to a point where the difference becomes significant at α. This was attributed to the fact that variance was not large enough as it would be expected from a sample of size NxK due to inherent dependency caused by overlapping training and test sets.

- The following variance correction was proposed while maintaining the degree of freedom for the t-test :

  $t = \frac{\bar{d}}{S*\sqrt{\frac{1}{NxK}+\frac{n2}{n1}}} \sim t_{NxK-1}$ where n_1 and n_2 are the size of the training and the test sets. This modification was empirically shown to have improved the paired t-test in terms of Type I error and it does not suffer from high Type II error [28].

Wilcoxon's signed ranks test is a non-parametric test that is used for pair-wise comparison. In this case, the null hypothesis is that the matched samples from the two independent groups (algorithms) come from a distribution whose median is 0 and symmetric around this median.

4.2.2 Comparing Multiple Algorithms

In order to compare multiple algorithms on the same dataset, ANOVA (parametric) and Kruskal-Wallis or Friedman (non-parametric) tests can be used, however these methods also assume independent samples and we are not aware of any modified versions that tolerate dependency.

In comparing multiple algorithms, the null hypothesis is that there is no significant difference between the algorithms. ANOVA compares two algorithms by comparing two estimates of the variance (σ^2) one of which is always a valid estimator while the other is valid only if the null hypothesis is true. The null hypothesis is rejected if there is a significant difference between the two variance estimates. The Kruskal-Wallis test is a non-parametric version of the ANOVA test and similar to Wilcoxon's signed ranks test. The null hypothesis is that independent samples from multiple groups come from a distributions with equals medians.

4.2.3 Parametric versus Non-Parametric Tests

On the other hand, a number of researchers warn against the unjustified use of parametric tests in comparing machine learning algorithms. Sanchez et al. [128] claim that according to their experience:

> *The predominance of t-tests in machine learning related experimental designs is not completely justified. t-tests are the most powerful option when the sampling distribution is gaussian. But, in many machine learning problems this assumption is not true.*

A similar argument is made by Luengo et al in [93] based on empirical evidence in comparing the performance of Neural Networks with varying levels of complexity. Based on 2-fold and 10-fold cross-validation experiments, they show that the samples did not satisfy all of the prerequisites of a parametric test. The authors specify three conditions before applying a parametric statistical test (ANOVA) for comparing multiple algorithms:

- Independency: the training sets in 10-fold cross validation are not mutually exclusive thus violating the independency condition since results on two test sets might be correlated if the training sets

were so much alike. In the 2-fold cross validation case, training sets are mutually exclusive within each iteration although they may overlap across iterations.

- Normality: the authors employ Kolmogorov-Smirnov test to assess normality of samples.
- Heteroscedasticity: the authors employ Levene's test to verify if the samples to be compared have equal variance.

Demšar studies the problem of comparing algorithms on multiple and possibly unrelated domains in [44]. The author recommends the Wilcoxon signed-ranks test for comparison of two algorithms on multiple datasets. The author states that if the assumptions of a t-test are satisfied, it is more powerful than a Wilcoxon signed-ranks test otherwise it should not be preferred. Demšar's paper compares algorithms on different datasets from different problem domains. Therefore, the independence assumption is not violated since there is no overlap across different datasets.

It should be noted that, in cross-validation settings on a single dataset, the independence assumption is always violated due to overlap in training sets either within one iteration (1x10-fold) or across different iterations (Nx2-fold). However, this effect is generally ignored in most canonical cross-validation followed by statistical testing practice. Best to our knowledge, the 10x10-fold cross-validation followed by a corrected repeated k-fold cv test is the only option for comparing two algorithms, which explicitly addresses the effect of dependence. As for non-parametric tests, we are not aware of any method that corrects for the effects of dependence.

4.2.4 Summary

In this section, we presented an overview of the current practice of evaluating and comparing the performances of machine learning algorithms. Table 4.1 summarizes the multiple aspects of the cross-validation based evaluation and hypothesis testing based comparison techniques. Type I error, power and replicability of a cross-validation / hypothesis testing scheme are all important factors in deciding which scheme to choose. The overlap between the random subsets of the dataset causes dependence

between the performance estimates as a result, the independence assumption for a statistical hypothesis test to be violated. When the independence assumption is violated, the hypothesis test has a higher Type I error. A 5x2-fold cross-validation introduces less overlap between the training and test sets compared to a 10-fold partitioning. The 5x2cv t-test and the combined 5x2cv F-test were developed to be used in 5x2-fold cross-validation settings. They have lower Type I errors than a regular 1x10-fold cross validation followed by a t-test. 1x10-fold and 5x2-fold cross-validation schemes generate only 10 estimates while a 10x10-fold cross-validation scheme generates 100 estimates which is more appropriate for an hypothesis test and also more replicable as far as the effects of random partitioning is concerned. A corrected repeated k-fold cv test was developed to be used in a 10x10-fold setting where a variance correction was introduced in order to compensate for the overlap across random partitions of data. This test has been shown to have a lower Type I and Type II errors compared to the 5x2cv t-test [28].

Approach	**Type**	**Compare Pairs or Multiple Algorithms**	**Cons.**	**Pros.**
(1) 1x10-fold cross-validation and paired t-test	Parametric	Pairs	High Type I error due to dependency (due to violation of independence assumption)	Uses 90% of data for training
(2) 5x2cv t-test [45]	Parametric	Pairs	Violation of independence assumption, high Type II error, sensitive to order to iterations, low replicability [28], underestimates performance due to use of 50% of data for training	Lower Type I error than (1)
(3) 5x2cv combined F-test [8]	Parametric	Pairs	Violation of independence assumption,underestimates performance due to use of 50% of data for training	Lower Type I error and higher power than (2), robust against the order of iterations
(4) 10x10cv and corrected repeated k-fold paired t-test [28]	Parametric	Pairs	Computationally more expensive (100 runs)	Corrected variance estimation to compensate for dependency, lower Type I and Type II error than (2), higher replicability [28], uses 90% of data for training, larger sample for a hypothesis test
(5) Wilcoxon Signed Ranks Test	Non-parametric	Pairs	Sensitive to ties, independence is always violated in cross-validation, no known version that tolerate this violation	More appropriate when the samples are not normally distributed
(6) ANOVA	Parametric	Multiple	Independence is always violated in cross-validation,no known variance correction such as in (4) exists	More powerful than non-parametric techniques when when the assumptions hold
(7) Friedman or Kruskal-Wallis Tests	Non-parametric	Multiple	Independence is always violated in cross-validation, no known version that tolerates this violation	More appropriate when the assumptions of a parametric test do not hold

Table 4.1: Summary of various cross-validation and statistical tests used in classifier comparisons

4.3 Pairing Feature Extraction and Classifiers

Running an feature extraction algorithm on the whole dataset before classification creates underestimated generalization error leading to overfitting. The consequence is more drastic in the case of a pre-processing step that uses the label information such as the multiple discriminants analysis (MDA) method. Computing the dimensionality reduction mapping on the whole dataset is similar to training and testing a classifier on the same dataset which might produce a mapping that does not generalize to unseen data. In order to avoid this condition, it is necessary to perform the feature extraction within the cross-validation coupled with the classifier.

For each training/test set pair, a mapping from the original to derived features are computed on the training set. Then the mapping is applied to the test set and the classifier is trained on the transformed training set and tested on the transformed test set (algorithm 3).

Algorithm 3: CombinedFEC: Pairing feature extraction/classification in NxK-fold cross-validation

foreach *Cross-Validation Repetition (r=1 .. N)* **do**
  foreach *Training/Test set pair (f=1 .. K)* **do**
    M=Learn_Transformation($tr_{r,f}$);
    $P_{tr_{r,f}}$=Performance(Classifier,M($tr_{r,f}$));
    $P_{ts_{r,f}}$=Performance(Classifier,M($ts_{r,f}$));
  end
end

Figure 4.6 shows a comparison of dimensionality reduction/classifier pairs on the olive dataset. For each training set, test set pair, the dimensionality reduction mapping is computed based on the training set and then applied to the test set. The classifiers are then evaluated on the transformed versions of the training set-test set pairs within each cross-validation iteration for 50 iterations. The plots show three classifier performance measures: accuracy, weighted AUROC and the weighted F-measure. The results indicate that, on the average, neither PCA nor MDA can produce 2D representations of the data such that classifier performance is maintained. In fact, for all classifiers, the plots reveal noticeable drop in average performance when dimensionality reduction is performed. Dimensionality reduction seems to have different affect on each classifier. Both PCA and MDA have drastic effect on the SMO classifier to a point where it ranks as the worse classifier after dimensionality reduction. Based on the wAUROC measure, the drop in performance seems less drastic than the other two measures. However, the

corrected repeated k-fold cv test ($\alpha = 0.05$) indicates that the difference is indeed statistically significant for all classifiers either with PCA or MDA.

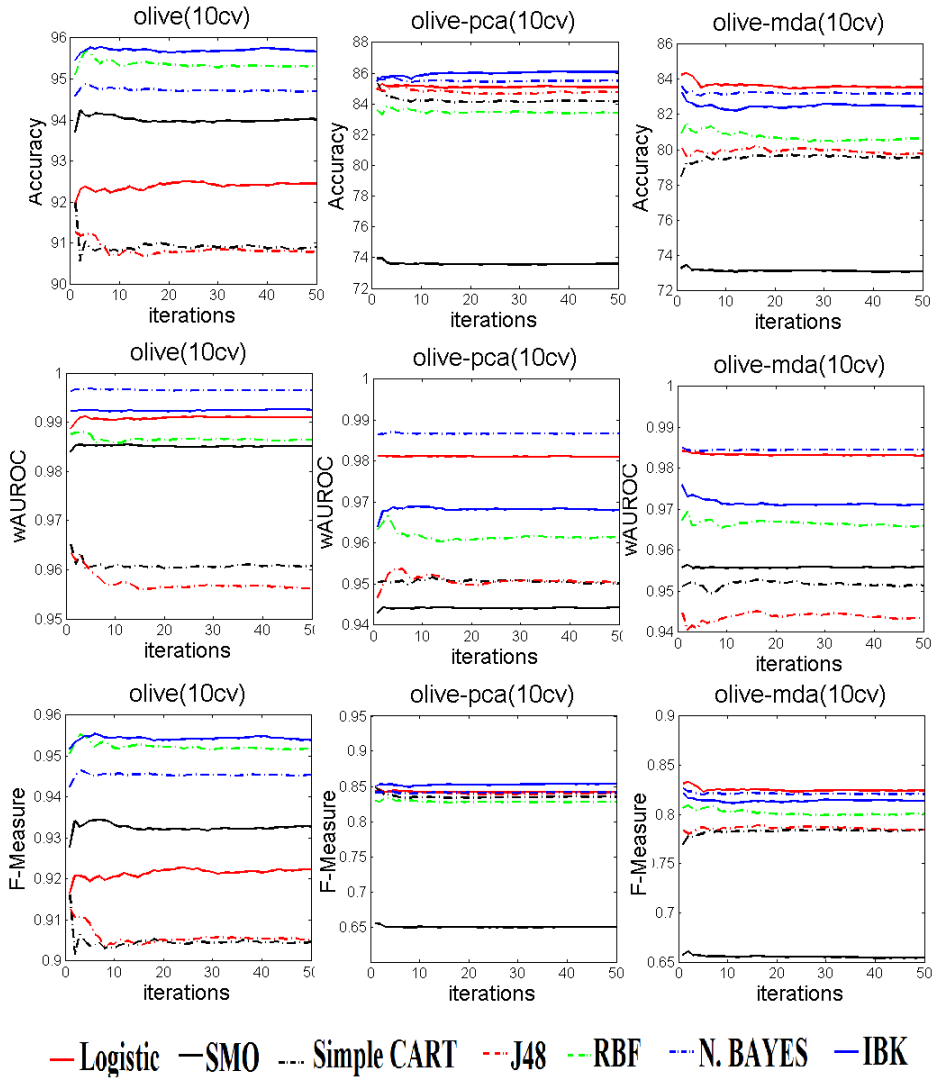


Figure 4.6: Comparison of the original attributes / classifier pairs to dimensionality reduction (2D) / classifier pairs on Olive dataset using 10-fold cross-validation up to 50 iterations. The classifier performance was better when all original features were used.

4.4 Cross-Validation in Genetic Programming Based Data Mining

The performance of a stochastic optimization technique on a problem is highly dependent on the initialization. In genetic programming, the randomly created initial population drives how the search pro-

gresses. Therefore, it has been a common practice to perform multiple runs in order to gather a stable estimate of the algorithm's performance across a number of initializations. In data mining experiments using a genetic programming technique, the use of cross-validation introduces more options to be considered. Some researchers adopt a 10-fold cross-validation scheme [140], while others prefer 5x2-fold cross validation [163, 84]. In each case, the researchers are forced to decide if more runs should be performed on each unique training set-test set pair or more iterations of cross-validation should be tried instead. Namely, the problem is to decide if we should choose more runs with different initial populations for one training set-test set pair or simply more runs with different training set-test set pairs while trying a small number of runs with different initial populations.

Regardless of the choice, the number of runs that is computationally feasible is limited, therefore it is important to select an experimentation scheme that returns better results. The published work seem to favor more variation in terms of input data rather than varying the initialization. In [140], the authors spare 2 runs per training set-test set pair in a 1x10-fold cross-validation scheme for a total of 20 runs on each dataset. In [84], the authors average the results of 5 runs on each training set-test set pair. They use 5x2-fold cross-validation, therefore their results were based on 50 runs on each dataset. In [163], the authors specify that they repeated 5x2-fold cross validation 10 times for a total of 100 runs for each dataset. However, it was not clear if it was implemented as 10 runs for each training-test set pair or as 50x2-fold cross-validation. In [69], we have reported our results based on a 10x10-fold cross validation scheme for a total of 100 runs (one run for each training-test set pair).

Figure 4.7 shows various sets of 100 MOG3P runs on BUPA and PIMA datasets. The results for Nx2-fold and Nx10-fold cross-validation runs are shown on the PIMA dataset and Nx10-fold cross-validation runs are shown for the BUPA dataset. The numbers in parentheses indicate the number of times the algorithm was run on each training-test set pair. The median test set accuracy is higher for 10-fold cross-validation settings than 2-fold cross-validation. Within 2-fold or 10-fold experiments, the difference in varying either the input data (more iterations) or the initial population (more runs per training-test set pair) is not significant according to an unpaired t-test at $\alpha = 0.05$.

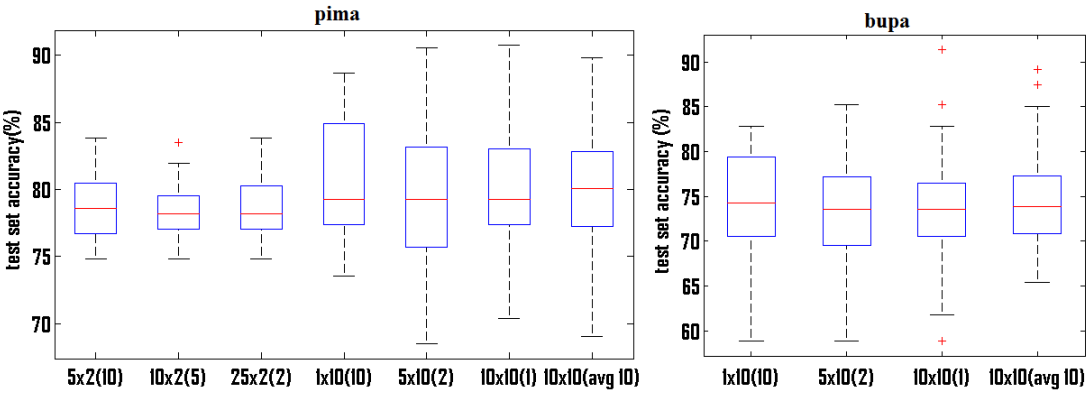


Figure 4.7: 100 runs of MOG3P on two datasets. For PIMA dataset, Nx2-fold and Nx10-fold cross validation results are shown. For BUPA dataset, Nx10-fold cross-validation results are shown. The numbers in parentheses indicate the number of times the algorithm was run on the same training-test set pair (varying initial population).

In our experiments, we chose to vary the input data (more cross-validation iterations) rather than sparing more runs on one training-test set pair. If we chose to perform more runs on each training-test set pair in order to smooth the effect of random initializations of the genetic programming algorithm, we would need to compute the average performance across all these runs and take it as the performance for that specific training-test set pair. Since we are using a paired statistical test to compare algorithms on the same training-test set pair, given that the number of total runs is fixed (such as 100), we would need to keep the number of iterations small. This would give us a smaller sample of performance estimates to compare algorithms. On the other hand, varying the input data by allocating more runs for different training-test set pairs gives us more performance estimates and a larger sample for a paired statistical test.

We conducted further experiments in order to validate our choice of varying input data rather than the initial population. We ran our algorithm 10 times per training,test set pair using the 10x10-fold cross-validation scheme for a total of 1000 runs. For each training,test set pair, we computed the aggregate performance by averaging the 10 runs which gave us 100 performance estimates (one for each training,test set pair). We call this scheme 10x10(avg 10)(figure 4.7). The only visible difference between the 10x10(1) and 10x10(avg 10) schemes (for bupa dataset) is the apparent increase in lower bound on the estimated test set accuracy when results of 10 runs are averaged for one training,test set

pair. This is not surprising, because it is known that more runs are needed in order to smooth out the effect of relatively unsuccessful performance depending on the initial population. However, this was not a consistent observation for all datasets. According to the corrected repeated k-fold cv test ($\alpha = 0.05$), there is no significant difference between the 10x10(1) and 10x10(avg 10) runs. We conclude that performing 1000 runs by varying both the initialization and the input data as opposed to 100 runs by varying the input data would bring unnecessary computational burden without statistically significant difference. Therefore, we prefer the 10x10(1) cross-validation scheme in our research.

4.5 Discussion

The goal of this section is to point out the importance of consistency across experimental set up in order to ensure reproducibility (or at least repeatability) of the reported results. A large number of researchers who are proposing a machine learning method based on genetic programming are utilizing the same benchmark datasets from the UCI repository. Generally, a NxK cross-validation scheme is the preferred technique for conducting experiments. Because of computational concerns, generally N is small (< 10) as K is generally 2 or 10. As we pointed out, it has been shown that the results would be highly unstable for such small number of iterations. Therefore, reproducibility is not guaranteed using a canonical cross-validation method. Due to computational concerns, it is not always possible to run a large number of cross-validation iterations until a stead-state is reached and guarantee reproducibility. For all experiments reported in this thesis, we used a standard set of random number seeds when randomizing the order of the data records for each cross-validation iteration. By eliminating the variation that could have been introduced by the ordering of the data we aim to ensure comparability of the experiment results such that any statistical difference is actually related to the algorithm only.

For our experiments, we create the stratified cross-validation partitions using the consistent set of random seeds and store the files so they can be used for all runs. This is especially important for a stochastic algorithm such as a genetic programming based algorithm like ours. When we compare the

effect of any modification on our algorithm, we need to make sure that the statistical difference between the modified and baseline versions are indeed due to the modification and not due to data ordering.

We presented an overview of various statistical tests that were used within cross-validation. Although it is known that at least the independence assumption of a t-test is not satisfied in a cross-validation setting, the researchers tend to ignore this violation and utilize a t-test especially for small number of iterations and folds (such as 5x2-fold). However, a corrected repeated k-fold cv test for the 10x10-fold cross-validation scheme which can be adopted to any NxK cross-validation scheme in order to compensate for the effects of dependency in the samples and reduce Type I error of a t-test. Best to our knowledge, no such corrected version of a non-parametric test exists for use in a cross-validation setting.

We also compared two options for setting up NxK cross-validation experiments for genetic programming based data mining experiments. Given that the number of total runs is limited, it is important to organize the experiments such that the likelihood of finding better models is increased. One option is to keep N small and perform more runs on each of the K training set,test set pairs and average over all runs on each pair. This approach aims to compensate for the effect of the randomly generated initial population. The alternative is to keep N large but to perform less runs on each unique training set,test set pair. We found no significant difference between varying the input data versus the initial population for a fixed number of genetic programming runs on the benchmark datasets. However, if we choose to vary the initial population by performing more runs for each training-test set pair, we would need to compute the average performance across all these runs in order to get one aggregate estimate for that training-test set pair. On the other hand, varying the input data by sparing more runs for different training-test set pairs gives us a larger sample of performance estimates. Therefore, we chose to run our algorithm for 100 times with more cross-validation iterations (different training-test set pairs) rather than more runs with the same input data. We also chose to utilize a 10-fold cross-validation scheme rather than a 2-fold cross-validation scheme since the algorithm achieved higher test set accuracy when the training set was larger (90$\%$ of the data as opposed to 50$\%$). As for the statistical significance test

when comparing two algorithms, we utilize the corrected repeated k-fold cv test proposed by Bouckaert and Frank in [28].

We refrain from comparing our results directly to published results that are using similar techniques. In most publications, the researchers report the mean and standard deviation of performance estimates of their algorithms. Even though it does not make sense to compare the reported mean values as a way to claim superiority of one's algorithm, we have seen a number of published papers using genetic programming for data mining tasks on selected UCI datasets presenting a table of those values they had picked from previous publications along with their results in order to claim superiority (for example: [163, 107, 140], all journal papers). In some cases, the authors acknowledge that the results may not be comparable due to differences in experimentation but they still claim that it gives an idea about how their algorithm performed compared to others. Based on our analysis we presented in this section, we believe that such a practice should be discontinued unless it can be assured that the results are indeed comparable.

The UCI dataset has been a great resource in providing a centralized repository for the machine learning community to compare their results. However, due to the experimental differences that we have pointed out in this section, there is a need for a standard way of reporting results so that the comparisons would make more sense. Therefore, we chose to make the datasets along with the cross-validation partitions and our raw results publicly available for any researcher who would like to compare their results to ours.

Chapter 5

Genetic Programming Projection Pursuit (G3P)

This chapter introduces our genetic programming based algorithm for interpretable dimensionality reduction for labeled datasets. In a general data classification scenario, there are two related goals: finding the best feature representation (feature extraction) and finding the most appropriate classifier for the problem. It is well known in machine learning literature that each classification algorithm has a model bias towards certain data characteristics and the classifier's performance depends on the degree of match between its bias and the data characteristic [77]. The problem of finding a feature representation/classifier pair for a problem domain can be attacked in various ways, such as:

- Find a feature representation based on a criterion (*pre-processing approach*): preserve variance in the data (PCA), preserve pair-wise distances (MDS, ISOMAP), minimize the ratio of intra-class to inter-class variation (MDA). Then train a classifier.
- Choose a classifier to use and find a feature representation that optimizes that classifier's performance (*wrapper approach*)
- Search for the best classifier/feature representation pair for the problem (*joint approach*)

We adopted a joint approach, where instead of one classifier, we consider a set of classifiers and search for a classifier/feature representation pair that is both *discriminative* as far the classification performance is considered and *interpretable* as far as the feature representation is considered. Since our interest is visual analytics, the feature representation we focus on is 2D scatterplot.

The goal of the algorithm is to find interpretable (simple) data transformations that map higher dimensional labeled data into a 2D representation (scatterplot) where the class structures are easily identifiable. While doing this we also aim to identify the most appropriate classifier from a small set of well-known classification algorithms. The algorithm is named Genetic Programming Projection[1] Pursuit (G3P)[2] since it searches for data visualizations which are easily interpretable both visually (visual interpretability) and in terms of the visualization axes (semantic interpretability) that are also discriminative (classifiability) feature representations for classification algorithms.

This section is organized as follows: first, we provide an overview of the related work that are closest to ours in terms of visual analytics and/or use of genetic programming in feature extraction for data classification. Then, we introduce our evolutionary multicriteria dimensionality reduction method namely the G3P. We present detailed discussions on each of the objective criteria in respective sections: classifiability (section 5.4), visual interpretability (section 5.5.2) and semantic interpretability (section 5.6). We also discuss the impossibility of enumerating the whole space of data transformations in section 5.7.6. Finally, we present the G3P experiment results and discuss its efficacy with respect to finding highly discriminative and easily interpretable data transformations for a number of data classification problems.

[1] Note that in linear algebra, a projection (P) is a (orthogonal or oblique) linear transformation from a vector space R^N to a vector space R^M where $P(P) = P^2 = P$. For data visualization, a linear projection from the vector space of original variables to a 2D vector space is a pair of linear expressions. However, linear projections are not the only options for data visualization. Any arbitrary function (linear or non-linear) of the original variables can be used to map the data onto 2D or 3D coordinates for visualization. In this thesis, we use the term projection as a general term meaning any arbitrary (linear or non-linear) data transformation (mapping) function.

[2] We note that the abbreviation G3P is also used for Grammar Guided Genetic Programming. However, the difference should be clear from the context.

5.1 Related Work

In this section, we overview related work in GP based feature extraction for classification problems. We categorized the algorithms with respect to *consideration of data visualization*, *feature extraction and assessment of discriminative power*, *feature interpretability* and *evaluation* aspects.

Visualization. Our method is at the intersection of the visual analytics and genetic programming research fields. Namely, our algorithm is an application of genetic programming into the field of visual analytics. In that regard, the closest work to ours in terms of explicitly addressing data visualization is presented by Valdes et al. in [151, 150]. The authors utilize an evolutionary computing method in order to generate 3D scatterplots for exploration of various medical datasets in a virtual reality environment where the accuracy of a nearest neighbor classifier (supervised) and a pairwise distance preservation measure (unsupervised) are used as the objectives to optimize simultaneously. They use a fixed length GP representation that is called Gene Expression Programming (GEP) in order to evolve three independent data transformation functions to generate a 3D visualization. The authors' choice of representation enforces a certain functional form: sum of five linear/nonlinear expressions of a predetermined size.

Although the hybrid approach of using supervised and unsupervised objectives for dimensionality reduction is an appealing idea, we note that the pairwise distance preservation measures (S stress, Sammon error) the authors incorporate are based on the Euclidean distance in the full space. It has been shown that Euclidean distance in high dimensional spaces can be meaningless due to curse of dimensionality [20], due to the fact that the distance to nearest data point approaches to the distance to the furthest data point as the dimensionality increases. It has been shown that this can happen for as small as 15 dimensions depending on the data. Thus, trying to preserve a flawed distance measure would give misleading results. Alternative measures to Euclidean measure in the full space can be constructed by searching for a subset of dimensions that would make the distance calculation 'meaningful' (such as in [6, 156]). In summary, it appears that in order to preserve a distance measure between the points for visualization purposes, we have to construct a meaningful distance measure in

the original space first which is by itself a computationally expensive search problem.

Feature extraction for Classification. The general problem of feature extraction for classification tasks have been addressed in a number of GP related papers. These methods consider the classification accuracy as their primary objective. In some cases, the number of evolved features is predetermined such as one *superfeature* [40], two or three in [53], whereas in other cases it can vary based on the problem (even more than the number of original variables in the dataset [163, 164]). The method proposed by Bot [27] makes use of a stopping criterion where new features are constructed only if they help increase the training accuracy. The author's findings on a number of benchmark datasets from the machine learning literature reveal that the number of features needed for good classification performance tends to increase as the number of classes increases. However, none of these techniques were specifically designed for data visualization except for the research presented by Valdes et al. [151, 150] as mentioned above.

Classifier selection and feature extraction have been jointly studied only in a few cases. Sherrah et al. propose a wrapper based feature extraction method [135], where each GP individual is associated with one specific classifier which can be either a generalized linear machine, a k-nearest neighbor or a maximum likelihood classifier. The fitness of each representation is then assessed by the accuracy of its assigned classifier. A similar approach is presented by Smith in [138] within the context of evolutionary feature extraction, where each GP individual is associated with either a decision tree (C4.5), Naive Bayes or a k-nearest neighbor classifier. All other wrapper based GP feature construction methods we review in this section, either target one specific classifier [84, 53, 27, 151, 150] or utilize each classifier separately in independent GP runs [138].

Measures other than accuracy of a trained classifier have also been utilized in order to assess the quality of the feature representations. Guo and Nandi utilize three class separation measures based on the Fisher Discriminant Analysis (FDA) for binary classification problems. Their technique aims to construct a single feature by transforming the dataset onto a 1D representation where the two classes are clearly separable. The work presented by Day and Nandi in [40] pursues the same idea of creating

one *superfeature* that optimizes class separation. The technique extends to multi-class problems where group membership of each data item is computed in terms of the Fisher criterion. The authors derive fitness as a binary string (bit vector) of hits (correctly classified) or misses, rather than one aggregate value of hits (accuracy). The technique also utilizes specialized mate-selection and crossover operators in order to select and hybridize behaviorally dissimilar individuals as an attempt to increase diversity and improve the convergence towards better feature representations. Neshatian and Zhang [109] and Muharram and Smith [106] utilize measures that are used in decision tree classifiers as the splitting criteria on the feature space. These techniques are not considered wrapper based since no actual decision tree is built in order to assess the quality of the feature representation.

Feature Interpretability Interpretability of the constructed features is addressed in seven of the twelve publications reviewed in this section. However, in most cases, the interpretability is associated with the issue of bloat control and handled by keeping the expression size under control without any further consideration of the contents from an interpretability perspective. Controlling the expression size during the evolutionary process is done in a variety of ways:

- Size Limit: Otero et al. [111], Muharram and Smith [106] and Estebanez et al. [53] utilize a predefined upper limit on the size of the expressions. The genetic operators are not allowed to procedure any expressions exceeding the limit. However, this approach has the disadvantage that the expression size grows uncontrollably until it reaches the limit and if the limit is too large, the expressions tend to be large and therefore complex due to bloating.

- Incorporating into the scalar fitness function: Sherrah et al. [135] incorporate the expression tree size and the number of attributes into the fitness function in a weighted linear combination. Smith and Bull [140, 139, 138] also incorporate the expression size into the fitness computation as a scaling factor to increase/decrease the effect of the accuracy.

- Separate criterion in a multi-objective setting: Zhang and Rocket [163, 164] utilize the expression size as a separate criterion in their multi-objective genetic programming technique.

Besides controlling the expression size during the evolutionary process, post-evaluation simplification techniques without hampering the classification performance have been explored by Smith [139]. The method presented by Valdes et al. [151, 150] utilizes a linear GP representation instead of a binary tree based representation where each feature transformation function is the sum of a pre-determined number of non-linear expressions of fixed size. Although the need for interpretability is very well motivated in GP based feature extraction, there appears to be no formal studies on how humans judge the interpretability of the expressions. Best to our knowledge, no user study such as the one we reported in chapter 3 has been reported.

Evaluation and Comparison. The most common performance assessment of the GP based techniques is the comparison of the classification performance on the GP induced feature representation to the performance on the original dataset. A number of papers also report comparisons to other feature extraction methods such as the PCA or Discriminant Analysis based methods [109, 61]. As far as the evaluation protocol is concerned, the practice varies though some form of cross-validation (such as 10-fold or 5x2-fold) is common with generally a t-test for significance testing. An overview of such techniques and a discussion on choosing a method for GP experiments have been presented in section 4.

In summary, table 5.1 presents the overview of the publications cited in this section. The methods are divided into three groups: the first five methods utilize measures that do not require a classifier to be trained on the data for feature extraction (filter based method), the second group of the methods utilize classifiers (wrapper based method) to evaluate the constructed features. The third group of methods (embedded methods) perform feature construction/selection and classifier induction simultaneously.

Citation	Type	Assessment of Discriminative Power	Targets Visualization?	Interpretability of transformations	Evaluation Method
Otero et al. (2003) [111]	Filter	Information gain ratio	No	Expression size limit	Comparisons to original features with C4.5 decision tree algorithm
Muharram and Smith (2005) [106]	Filter	Information gain, Gini index, χ^2 measures	No	Expression size limit	Comparisons to original feature set with C5, CHAID, CART decision tree algorithms and a neural network classifier
Guo and Nandi (2006) [61]	Filter	Three variants of Fisher Discriminant Analysis (FDA) class separation measures	No	None	Comparisons to PCA, FDA dimensionality reduction methods with MDC, MLP and SVM classifiers
Neshatian and Zhang (2008) [109]	Filter	Information entropy measure	No	None	Comparisons to original features and PCA dimensionality reduction with J48 (C4.5) decision tree algorithm
Day and Nandi (2011) [40]	Filter	Creates one *superfeature* that is assessed by 1D Fisher criterion (ratio of inter-class variance to intra-class variance)	No	None	Reports comparable results to Support Vector Machine and Multilayer Perceptron (MLP)
Sherrah et al. (1997) [135]	Wrapper	Accuracy of a randomly assigned classifier amongst generalized linear machine, k-nearest neighbor and maximum likelihood classifier	No	Scalar objective (expression size, number of features)	Comparisons to original dataset with generalized linear machine, k-nearest neighbor and maximum likelihood classifiers
Bot (2001) [27]	Wrapper	Accuracy of k-nearest neighbor classifier	No	None	Comparisons to original feature set with k-nearest neighbors and evolved features with k-nearest neighbors, minimum distance and parallelepiped classifiers
Krawiec (2002) [84]	Wrapper	Accuracy of J48(C4.5) decision tree classifier	No	None	Comparisons to original feature set with J48(C4.5) classifier
Smith and Bull (2005) [140, 139, 138]	Wrapper/Filter	Accuracy of J48(C4.5) decision tree classifier, Naive Bayes and k-nearest neighbor classifiers separately or randomly assigned to each GP individual	No	Scalar objective (expression size) and Post-run simplification	Comparisons to original feature set with J48, Naive Bayes and k-nearest neighbor classifiers
Estebanez et al. (2007) [53]	Wrapper	Accuracy of simple perceptron classifier	No	Expression size limit	Comparisons to original feature set with SVM, neural network and simple logistics classifiers
Valdes et al. (2007) [151, 150]	Wrapper	Accuracy of k-nearest neighbor classifier	Yes. Preservation of pairwise distances via S Stress and Sammon Error measures	Fixed sized and constrained representation	No comparisons to other classification techniques reported
Zhang and Rockett (2009,2011) [163, 164]	Wrapper	Multiobjective (misclassification error of Fisher Linear Discriminant(FLD), Bayes Error)	No	Tree complexity (expression size)	Comparisons to original feature set with a number of classifiers
Muni et al. (2006) [107]	Embedded	Accuracy	No	Ties are broken in favor of smaller expressions	Other results from the literature
Lin et al. (2008) [92]	Embedded	Accuracy	No	Number of features used	None

Table 5.1: GP based feature extraction (feature construction/selection methods

5.2 Multi-Objective Dimensionality Reduction for Visual Data Classification

We define problem of generating 2D scatterplots from a higher dimensional dataset as the task of mapping the original data features into two discriminative and interpretable features as follows:

Definition 2. Multi-Objective Dimensionality Reduction for Visual Classification

Let **D** be a labeled dataset given as a NxR matrix where N is the number of dimensions (variables, features, attributes) and R is the number of items (observations or records). A data model **M** is a TxR matrix that is the transformed form of **D** using **T** symbolic expressions where **T**=$\{2\}$ for visualization.

Let **F** be a function that returns a measure of how good a model **M** is, given classifiability criteria **C**, visualization criteria (**V)** and semantics criteria (**S**). Then, data model selection is the *process of selecting* the best model: argmax_i **F**($\mathbf{M}_i|\mathbf{D}, \mathbf{C}, \mathbf{V}, \mathbf{S}$).

Note that in our definition, we use the terms variable, feature, attribute and dimension interchangeably. The term variable is preferred in the statistics field while the term attribute is generally used in data mining. The pattern recognition community prefers the term feature since the raw data of any kind is converted into a vector representation using domain specific feature generation methods. In general, each dataset that is studied in machine learning is in a matrix form where each column is a dimension. The dimensionality reduction field studies various methods to reduce the number of columns in order to optimize the outcome of a machine learning task. Likewise, we use the terms data item, observation (statistics) and record (data mining) interchangeably in order to identify the rows of the data matrix.

5.3 Why Genetic Programming for Projection Pursuit?

Our algorithm is named Genetic Programming Projection Pursuit (G3P) since it is a GP based automated method to construct data transformations that would reveal *interesting* structures similar to exploratory projection pursuit [58, 148]. In projection pursuit, the 'interestingness' is assessed using a function (projection index) that is defined based on various properties of the data such as the density, skewness, kurtosis (unsupervised) and class separation (supervised). Hill climbing based optimization techniques are by far the most popular methods for optimization of the projection indices where other methods such as simulated annealing have also been explored [91].

In our research, we define interestingness by means of classifiability, visual and semantic interpretability of the data transformations. The definition of multi-objective dimensionality reduction that is given above covers how a data transformation is evaluated thus it acts as a goodness-of-fit measure. It is independent from the process of how the data transformation functions are generated. We adopted a population based stochastic optimization framework for our task which provided us with a principled way of creating and evaluating a large number of data transformation functions. As the data representation in this optimization framework, we chose the expression tree based genetic programming representation for the following reasons:

- We aim to evolve arbitrary functions that can be used as data transformation functions and expression trees are the natural representations where we can specify which operators and operands can be used to build a tree (functional symbols). Since we do not constrain the size and structure of a transformation function, GP provides us with a flexible representation to explore the space of possible transformations. GP can also allow us to incorporate any functional form through expression grammars.

- The interpretability (readability, complexity) of a transformation function can be directly inferred from the expression tree that is being evolved and also can be easily converted into the linearized form for humans to inspect.

- Since the genetic operators are directly manipulating the expression itself, it is easier to observe how they affect the data transformation function and the visualization that is being generated.

However, a variable-length representation such as GP suffers from the bloat (code growth) phenomenon (section 2.4.3.2). In our research, we pay extra attention to this shortcoming and incorporate various ways to prevent bloat (sections 5.9.3 and 6.2).

GP has been employed within the context of projection pursuit by Rodriguez et al. in [125]. The authors implement an expression tree based GP method that evolves combinations of a number of projection indices. Within the evolutionary process, each evolved index is utilized in a separate hill climbing based projection pursuit stage that creates a lower dimensional feature representation. During the projection pursuit stage, the number of features are not constrained to 2 or 3 for visualization, new features are sequentially added until a stopping criterion is met. The quality of the lower dimensional representation is then assessed in terms of the accuracy of a K-nearest neighbors classifier. In this technique, the new features are generated via a set of orthogonal linear combinations of the original variables while the optimization criterion (projection index) is evolved using the GP approach. This approach can be seen as a wrapper around the classical projection pursuit algorithm. Our approach is fundamentally different than this technique since we use GP to directly evolve data transformation functions which are not necessarily linear or orthogonal. We then assess the quality of the generated feature representation in terms of three main criteria: classifiability, visual interpretability and interpretability of the data transformation functions.

5.4 The G3P Algorithm

In G3P, we adopt a generational GP approach and each GP individual contains two expression trees that are evolved independently. Our implementation is based on a modified version of the open-source ECJ toolkit (version 20) developed by Luke et al. [94]. Algorithm 4 outlines the workflow of G3P.

Algorithm 4: The G3P algorithm for visual data classification

Input : NxR data matrix D
Set of functional symbols B =$\{b_1, b_2, ..., b_K\}$
Population size I
Fitness Function F
Classifiability Criterion C
Visualization Criterion V
Semantic Criterion S
Number of transformation expressions in T(2)

Output: NxR transformed data matrix M, data transformation expressions T

Randomly create initial population of transformation functions (expressions) T
foreach *transformation T in population* **do**
  compute M by applying transformation T to D
  compute $F(M|D, C, V, S)$ as the fitness of the transformation T
end
repeat
  Select the transformations (T) from the population as parents
  Perform breeding (crossover, reproduction, mutation operations) to create offsprings
  foreach *transformation T in population* **do**
    compute M by applying transformation T to D
    compute $F(M|D, C, V, S)$ as the fitness of the transformation T
  end
until *maximum number of generations are completed or an ideal solution is found*

As figure 5.1 shows, the algorithm assesses classifiability and visual interpretability on the phenotype of each individual, while semantic interpretability is assessed on the genotype.

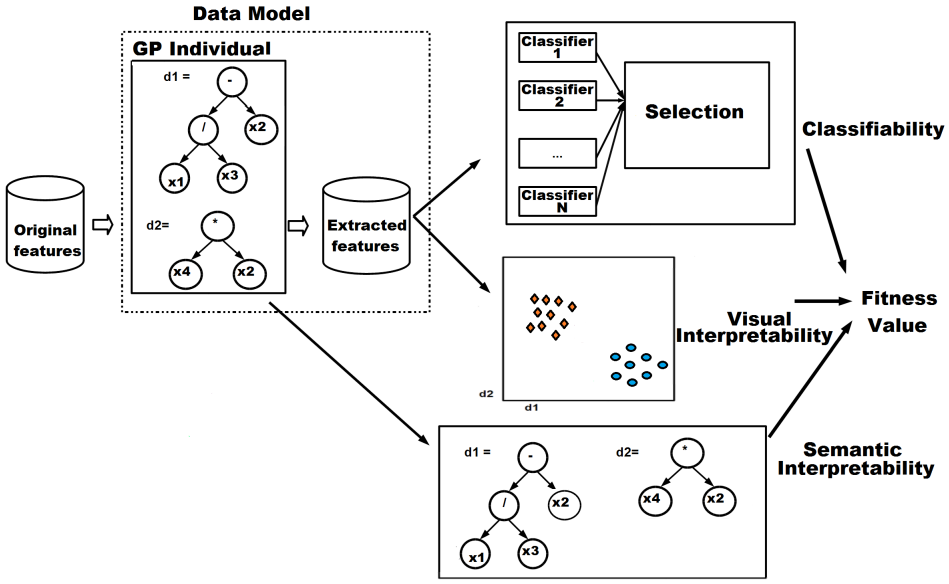


Figure 5.1: G3P diagram

We evaluate the G3P algorithm in a NxK cross-validation scheme (N=10,K=10) for a total of 100 runs where each G3P run works on one training fold and the performance of the best of run individuals are evaluated on the corresponding test set that was not used by G3P (algorithm 5). This evaluation protocol has been selected based on our study presented in chapter 4.

Algorithm 5: Evaluation of G3P in NxK-fold cross-validation

```
foreach Cross-Validation Repetition (r=1 .. N) do
    foreach Training/Test set pair (f=1 .. K) do
        T=G3P(tr_{r,f});
        P_{tr_{r,f}}=Performance(Classifier,M(tr_{r,f}));
        P_{ts_{r,f}}=Performance(Classifier,M(ts_{r,f}));
    end
end
```

5.5 Objective 1: Classifiability of the Constructed Feature Representations

The G3P algorithm has two goals: finding *highly discriminative* and *interpretable* 2D feature representations for the given higher dimensional dataset. Interpretability objectives will be covered in the next two sections. In this section, we concentrate on the measures of discriminative power of a feature representation and present how G3P evaluates the classifiability of the GP individuals.

5.5.1 Measuring Classification Performance

In a classification problem, a classifier's performance on unseen data is computed based on four indicators. Specifically, in a binary classification problem (class labels: positive and negative):

- *True Positive* (tp): Number of times a positive data item was classified positive
- *False Positive* (fp): Number of times a negative data item was classified as positive
- *True Negative* (tn): Number of times a negative data item was classified as negative
- *False Negative* (fn): Number of times a positive data item was classified as negative

Classifier accuracy (rate of correct classification) is defined as : $\frac{(tp+tn)}{total}$ where total=$tp+fp+tn+fn$ is the number of classified items. Similarly, error rate is defined as: $\frac{(fp+fn)}{total}$ and $accuracy = 1 - \textit{error rate}$. In optimization schemes the goal is to either maximize the accuracy or minimize the error rate of a classifier. Based on these four values, various other classifier performance measures can be defined (such as precision, recall, F-measure, Area Under ROC), however accuracy/error-rate are the simplest

measures to compute and the most generally utilized measures in reporting experiment results. The calculation of accuracy can easily be generalized into multi-class classification problems (accuracy: number of times the predicted class label matches with the true class label).

In GP based data mining literature, the scalar accuracy/error rate value is the most commonly used performance indicator. Notable exceptions are the representation of accuracy/error rate as a bit vector in [40] and class-specific decomposition of error rates in [114].

We called this objective the classifiability objective in order to indicate that we do not wish to tie the feature extraction process to one particular classifier that is fixed through the G3P runs. Rather, our goal is to find a feature representation that can be classified with high accuracy and then answer the question: if it can be classified with high accuracy, which classification algorithm should we use from now on? This issue can be addressed in a number of ways:

- Consider one data transformation and randomly selected classifier pair at a time and take the classifier performance on the transformed data as the classifiability measure
- Take one data transformation and consider an aggregate performance measure of a set of classifiers on the transformed data:
  - Minimum classifier performance: performance of the least successful classifier (worse classifiability)
  - Maximum classifier performance: performance of the most successful classifier (best classifiability)
  - Mean, Median classifier performance: typical classifiability

In each of the aggregated scheme outlined above, the classifier that will be associated with the data transformation is the one with best performance on the transformed data. The random classifier assignment per individual approach has also been utilized Sherrah et al. in [135] and by Smith in [138]. In Smith's approach, a special mutation operator was designed in order to replace an individual's randomly assigned classifier with another classifier.

5.5.2 Evaluation of Classifiability in G3P

In G3P we implement two options for evaluating the classifiability objective. We modify the GP Individual to include a classifier object. Every individual is assigned a random classifier at the initialization state. When a new a new individual is created during crossover, we assign its classifier according to a set of predefined options. Algorithm 6 summarizes this process.

Algorithm 6: G3P classifier assignment (ComputeClassifiability)

Input : Classifiers: set of active classifiers, Transformed Dataset: data, folds: #of folds
if *assignment_option=RANDOM_CLASSIFIER* **then**
  if *crossover_option=RANDOM_CLASSIFIER_SELECT* **then**
    individual.classifier=RandomlySelectOne(Classifiers) ;
    else if *crossover_option=RANDOM_CLASSIFIER_WITH_PARENTS* **then**

$$individual.classifier = BestOf(RandomlySelectOne(Classifiers), individual.parent1.classifier, individual.parent2.classifier);$$

    end
    individual.classifiability=EvaluateClassifierCV(individual.classifier,data,folds);
  end
  else if *assignment_option=ALL_CLASSIFIERS* **then**
    switch *aggregate_option* **do**
      case *MIN*
        individual.classifiability=Min(EvaluateClassifierCV(Classifiers,data,folds));
        break;
      case *MAX*
        individual.classifiability=Max(EvaluateClassifierCV(Classifiers,data,folds));
        break;
      case *MEAN*
        individual.classifiability=Mean(EvaluateClassifierCV(Classifiers,data,folds));
        break;
      case *MEDIAN*
        individual.classifiability=Median(EvaluateClassifierCV(Classifiers,data,folds));
        break;
    endsw
    individual.classifier=BestOf(Classifiers);
  end
end

The ALL_CLASSIFIERS option simply runs all classifiers on the individual data transformation and assigns the aggregate (MIN,MAX,MEDIAN,MEAN) performance value as the classifiability objective. The RANDOM_CLASSIFIER option assigns a randomly selected classifier to each individual at initialization time.

During crossover, we utilize one of two options based on the value of the classifier selection option. The option RANDOM_CLASSIFIER_SELECT assigns a random classifier to the offspring and RANDOM_CLASSIFIER_WITH_PARENTS assigns a random classifier to the offspring and then selects the best performing classifier amongst the individual's own classifier and the parents' classifiers.

5.6 Objective 2: Visual Interpretability

The essence of exploratory data visualization is the search for *interesting* data views. The search can be manual or automated. However, given the state of data mining today, it is no longer feasible to perform manual exploration. Therefore, it is important to develop automated measures that will help us discover useful data views based on the data mining task that we are performing. Since, the context of this thesis work is data classification, we are interested in exploring labeled higher dimensional data. In a data classification setting, the users are interested in those views of the dataset that show clear separation between the different groups in the data. In the machine learning and visual analytics literature, various measures have been introduced in order to capture this seemingly simple goal. An overview of these methods have been presented in section 2.3.

The issue that is related to using an automatic measure as opposed to manual exploration is the degree of match between the human perception and the automated measure. This is an important research topic in the visual analytics field and a number of user studies were reported in related literature. In chapter 3, we review related work and present our own user study in which we studied a large number of automated measures that can be used to assess visual quality of 2D scatterplots of labeled datasets. Based on our empirical study on four datasets, we found that classifier accuracy on a 2D view of the data can be an indicator of how the user will rate goodness of that view. Cluster validation indices from the machine learning literature demonstrated mixed results. Those measures that favored round shaped clusters rated the quality of those views that displayed different cluster shapes as low quality while the humans tend to rate them as higher quality views. However, the automated measures are in alignment with the human judgement if the view displays round shaped clusters. In summary, we found that all measures we investigated could be used to assess visual quality though it does not appear that one single measure (or a linear combination of the measures) outperforms all others in matching the human perception closely. One more highlight from our study was that, generally classifiers did not have a problem in classifying those views where members of the same class would be clumped

across different areas of the view, while the humans did not favor such views as much as the classifiers did. Therefore, we infer that humans tend to favor compactness as well as separation. Based on this observation, it makes sense to utilize such visual quality measures along with the classifiers in order to find such visualizations that are both easy to classify and visually preferable for the users.

In G3P, we utilize the classifiers under the classifiability objective and the visual quality measures (C Index, Davies-Bouldin Index, LDA Index, Class Consistency Measure, 2D-Histogram Density Measure) are considered as the explicit visual interpretability measures.

5.7 Objective 3: Semantic Interpretability of the Visualization Axes

Semantic interpretability of the visualization (a 2D scatterplot here) is the *meaning* of the visualization in terms of how the original variables relate to the visualization axes. In 2D case, each of the two visualization axes is defined by an arbitrary data transformation function which can take any form.

From a genetic programming viewpoint, the focus is the evolved expression and semantics of an evolved program (genotype) is tightly connected to the phenotype, namely its behavior. In fact, in symbolic regression literature, the researchers use the term semantic to indicate the output that the evolved expressions create [149]. Similarly, when genetic programming is used to create data visualizations, the semantics of an expression can be defined in terms of the visualization it generates. However, our focal point in this thesis is the visualization and we define semantic interpretability as the *semantic interpretability of the visualization*. The same point of view is also shared by Valdes et al. in [151], where they also present a dimensionality reduction method based on evolutionary computing. The authors indicate that the feature mapping functions provide the semantics that tie the original variables to the visualization axes. Figure 5.2 summarizes the difference between the two viewpoints on how semantics is defined. Though both approaches are completely valid, we choose the visual analytics point of view, therefore, when we mention semantic interpretability, it should be understood as the meaning of the visualization that is defined in terms of the data transformation functions evolved.

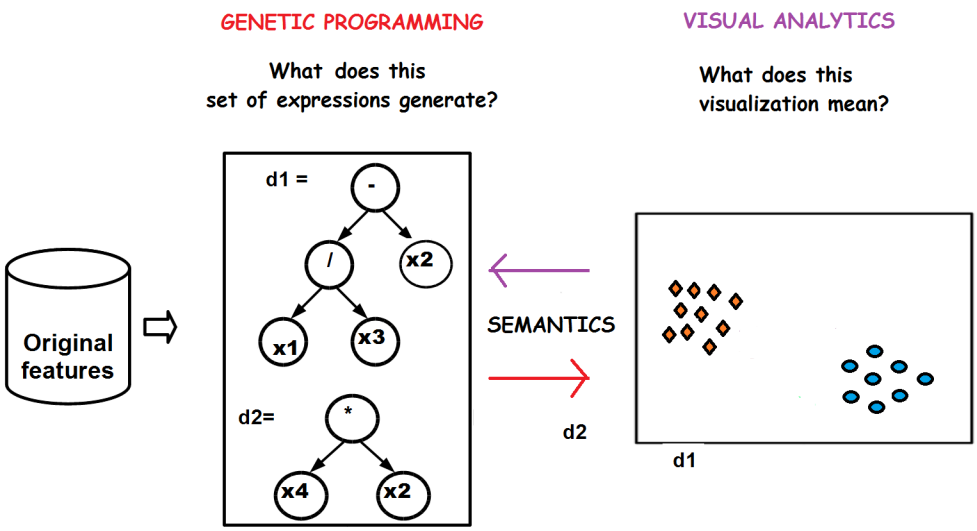


Figure 5.2: The term semantic in genetic programming and visual analytics contexts

Aside from the fact that complex expressions consume more computational resources and they tend to be too unstable for predictive modeling (generalization problem), they are also more difficult to read and interpret. In genetic programming literature, two different notions of expression complexity have been introduced: genotype-based and phenotype-based. Genotype-based methods are concerned with the structural properties (syntax) of the expression while phenotype-based methods consider the properties of the output generated by the expression.

The pheonotype-based measures of expression complexity have been proposed for the symbolic regression problems within the genetic programming literature, where a response variable is estimated based on a number of independent variables. Imada and Ross present a feature based approach for the symbolic regression problem [70]. The authors compute 17 statistical and shape based features taken from the time series modeling literature (such as self-similarity, periodicity, nonlinearity) on the output of each evolved expression. Although the authors do not present these features explicitly as complexity measures, they can be used to model expression complexity as well. Vladislavleva et al. present the *order of nonlinearity* as complexity measure in [158] for the symbolic regression problem domain. The authors advocate that nonlinearity increases complexity of the expressions thus making them difficult to compute and interpret. In this context, interpretability is linked to the smoothness of the response

surface generated by the evolved expression. The authors suggest that highly nonlinear surfaces indicate complex and unstable models for prediction. In their approach, each subtree is converted into a Chebyshev polynomial approximation with minimal degree and the complexity of the whole expression is computed in terms of the complexities of these polynomials. The authors utilize the order of nonlinearity measure in conjunction with the expression complexity measure (by alternating the complexity measure per generation). They report improved results in terms of accuracy and compact expressions with smoother response surfaces. The order of nonlinearity measure is a principled approach that is useful in regression modeling, however it is computationally expensive. Another similar measure that tries to model the smoothness of the function has been proposed by Vanneschi et al. in [152]. This measure computes the total curvature of a multi-dimensional function as the sum of the curvatures of its 1D components.

Though our problem (dimensionality reduction) is related to symbolic regression, such phenotype-based measures are not directly applicable. In our case, the visual interpretability measures we defined in the previous section serve as phenotype-based complexity measures related to the evolved expressions. In this section, we concentrate on the genotype based measures of expression complexity.

This rest of this section presents five measures we utilized in order to characterize the complexity of the data transformation functions. Total Tree Size (TS) is simply the total number of nodes in the expression tree which is the most commonly used complexity measure across the genetic programming literature in order to control bloat and increase readability of the expressions. In order to increase readability, simplification of the best-of-run expressions has been proposed. For instance, in [139], the evolved feature construction expressions are simplified by eliminating redundant subtrees. Enforcing a certain functional form on the evolved expressions is also a technique that can be used to increase interpretability. In [103] grammar based canonical functional forms are utilized in order to evolve interpretable solutions to circuit modeling problems.

The second measure, expression complexity was presented in [158] where the complexity of each node is defined as the sum of its subtree sizes. We introduce three complexity measures: the *weighted*

expression complexity measure is based on the expression complexity where each operator is assigned a weight with respect to its nonlinearity, *structural complexity measure* is based on our user study presented in section 3.3 where we define complexity in terms of tree size, tree depth and in terms of the identifiable blocks in the expression, the *weighted structural complexity* applies weights to the operators with respect to their nonlinearity. We group the operators into classes with respect to their complexity and assign weights for each kind of operator. Less complex operators are assigned smaller weights. For instance; the addition and subtraction operators are linear and they are considered less complex than the division or multiplication operators. All five measures presented in this section assign larger values to more complex expressions, therefore in G3P they are minimized.

5.7.1 Total Tree Size (TS)

Given an expression, the total tree size (TS) measure is simply computed as the number of nodes appearing in the expression : $I_{TS}(n_i) = 1 + \sum_{c=1}^{\#children(n_i)} I_{TS}(n_{ic})$

This is the simplest measure which does not consider any other structural property of an expression. Moreover, each node regardless of being an operator or an operand has the same contribution to the expression's complexity (figure 5.3).

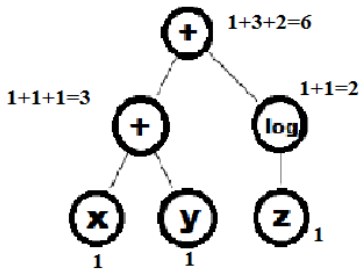


Figure 5.3: Computation of the Tree Size (TS) Measure

5.7.2 Expression Complexity (EC)

The expression complexity measure introduced in [158] is based on the tree size measure. The complexity at each node is computed as the sum of the complexities of all subtrees as follows:

$$I_{EC}(n_i) = I_{TS}(n_i) + \sum_{c=1}^{\#children(n_i)} I_{EC}(n_{ic})$$

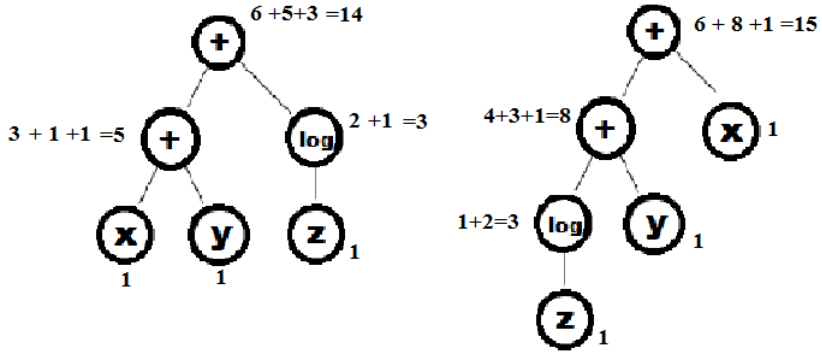


Figure 5.4: Computation of the Expression Complexity (EC) Measure

This measure assigns smaller complexity values to balanced trees representing relatively symmetric expressions (figure 5.4).

5.7.3 Weighted Expression Complexity (WEC)

Based on the expression complexity approach, we consider each operator separately and assign a weight according to its linearity/nonlinearity. The multiplication, division, logarithm, power operators are nonlinear whereas the addition and subtraction operators are linear. We call this measure the *weighted expression complexity measure* and compute it as follows:

$$I_{WEC}(n_i) = fc(n_i) * I_{TS}(n_i) + \sum_{c=1}^{\#children(n_i)} I_{WEC}(n_{ic})$$

where $fc(ni)$: functional complexity coefficient of the operator

In our experiments we assigned the following values as the functional complexity coefficients: constants and variable symbols:1, min,max: 2, addition, subtraction:3, log,sqrt:4, multiplication, division, power:5. An expression is considered more complex if the nonlinear operators are closer to the root of the expression tree. Balanced expressions are still considered less complex (provided that they have the same set of operators) as in the expression complexity measure (figure 5.5).

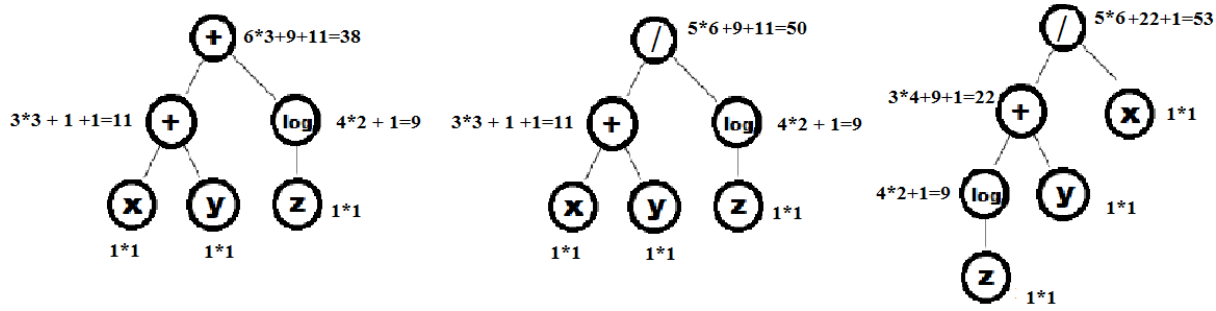


Figure 5.5: Computation of the Weighted Expression Complexity (WEC) Measure

5.7.4 Structural Complexity (SC)

In section 3.3, we presented an empirical study that aimed to understand how humans would rate interpretability of a given mathematical expression with varying levels of complexity in terms of its structure. We had hypothesized that certain structural characteristics would make it easy or difficult for humans to comprehend an expression. Based on our user study, we had came to the conclusion that, though size was an important factor, tree depth, the number and size of identifiable blocks (subtrees) are also related to human interpretability of an expression. Specifically, we found that longer size and higher tree depth makes the expressions difficult to interpret while existence of compact blocks improve interpretability. We call this measure the structural complexity of an expression tree and define it as follows:

$$I_{SC}(n_i) = \frac{I_{TS}(n_i) * depth(n_i)}{\text{\#Blocks } (n_i) * \text{ Avg. Block Size}(n_i)} + \sum_{c=1}^{\#children(n_i)} I_{SC}(n_{ic})$$

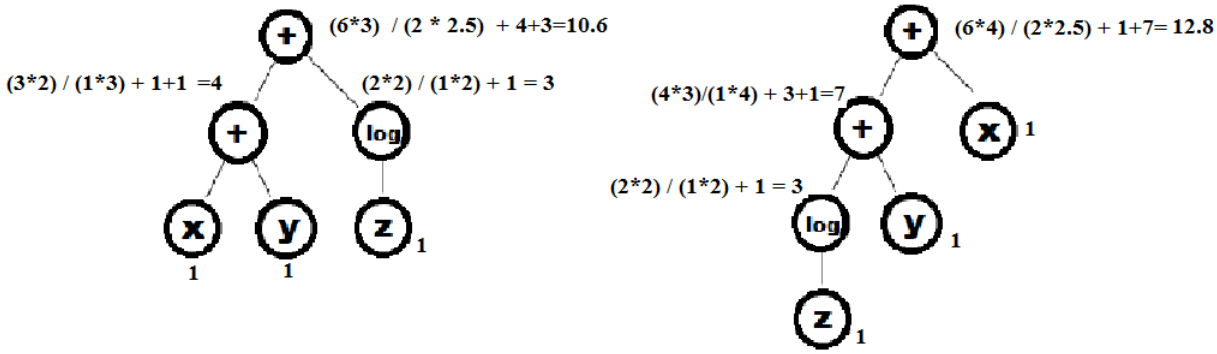


Figure 5.6: Computation of the Structural Complexity (SC) Measure

As in the EC measure, balanced trees (thus relatively symmetric expressions) are considered less complex (figure 5.6).

5.7.5 Weighted Structural Complexity (WSC)

We define a weighted version of structural complexity as follows:

$$I_{WSC}(n_i) = fc(n_i) * \frac{I_{TS}(n_i) * depth(n_i)}{\text{\#Blocks } (n_i) * \text{ Avg. Block Size}(n_i)} + \sum_{c=1}^{\#children(n_i)} I_{WSC}(n_{ic})$$

where $fc(ni)$: functional complexity coefficient of the operator

The functional complexity coefficients for the operators are assigned similar to the WEC measure presented above. Likewise, an expression is considered more complex if the nonlinear operators are closer to the root of the expression tree and between two expression trees with the same set of symbols, the balanced expression tree is assigned a lower complexity value (figure 5.7).

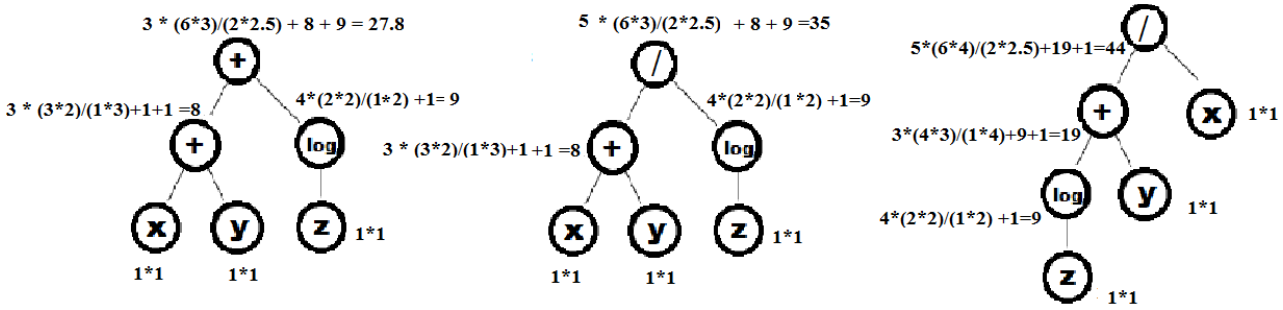


Figure 5.7: Computation of the Weighted Structural Complexity (WSC) Measure

5.7.6 Summary

We have presented five expression complexity measures, two of which (TS and EC) have been introduced in the genetic programming literature. We then introduced three complexity measures. One measure (SC) is based on our user study on how humans perceive complexity of algebraic expressions. Based on the empirical results, we defined a structural complexity measure based on the tree size, tree depth, the number and sizes of identifiable blocks (subtrees). We also considered the content of the expressions as well as the structure. We introduced weights to be assigned to the operators based on their relative complexity. In our experiments the weights were chosen in a way that linear operators such as addition and subtraction are considered simpler than the nonlinear operators such as multiplication, logarithm and division. By introducing the operator weights, we hypothesized that types of

operators would have an effect on user preference, however our user study was inconclusive in this regard. The users seemed to have been more concerned about the readability of the expressions than the properties of the individual operators. Nevertheless, in a data mining setting, the users would have been concerned about the operator complexity as well, therefore we believe that incorporating weights according to operator types is helpful.

Figure 5.8 presents examples of expression trees along with their complexity values.

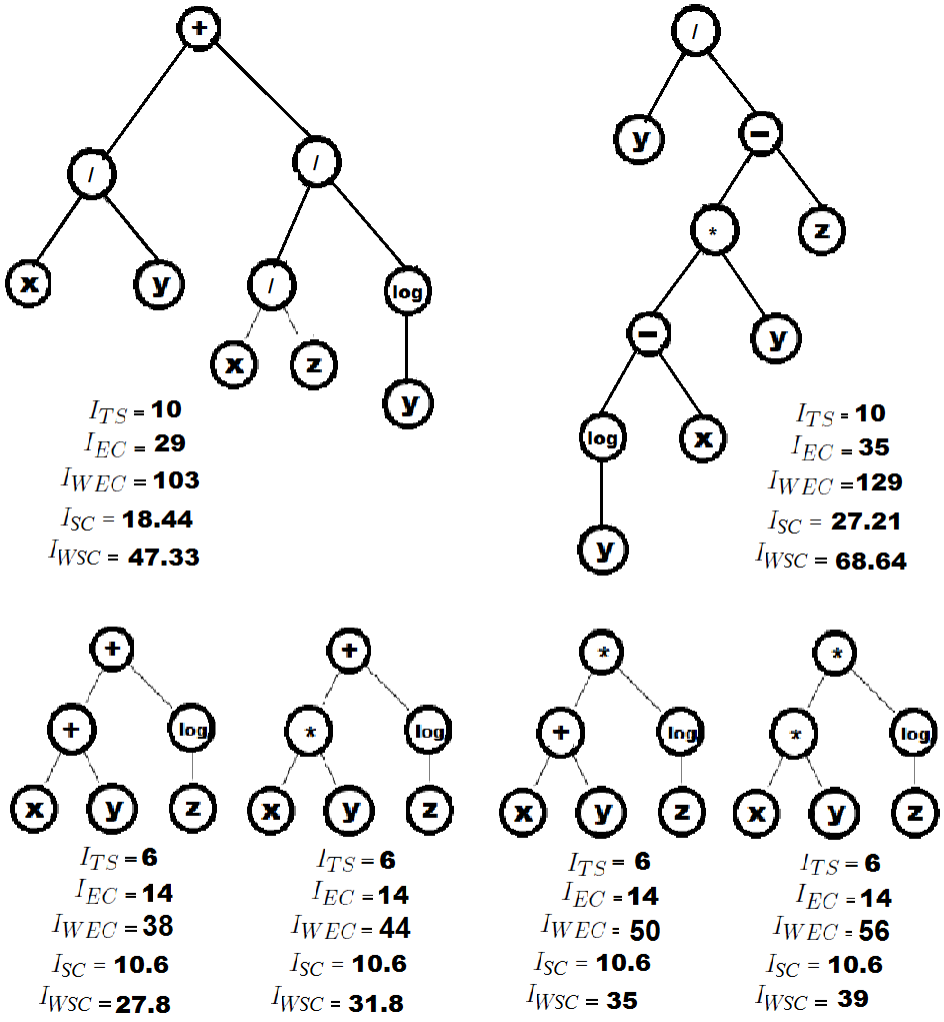


Figure 5.8: Example expression trees and complexity measures

The bottom row contains four similar trees showing that expressions containing nonlinear operators receive higher complexity values under the weighted (WEC and WSC) schemes. We also stated that

except from the TS (tree size) measure all other complexity measures favor balanced expression trees. All measures are based on syntactical properties of the expressions, however by discriminating between the types of the operators, we aim to incorporate clues on functional characteristics of the expressions as well.

In genetic programming based data mining tasks, complexity of the evolved expressions are of interest for three main reasons:

- Reduce bloat (code growth) and computational requirements: If the fitness values of two evolved programs are the same, then the one with the smaller expression tree is preferred. We note that in the case of weighted complexity schemes we defined above, it is not guaranteed that a less complex expression tree is also smaller. Thus they can not be used to strictly control bloating since bloating is measured based on the tree size.

- Increase human interpretability: All measures presented above aim to increase human interpretability from different perspectives (reducing size, balancing the tree, increasing amount of easily identifiable blocks, preferring simpler operators over complex ones).

- Increase generalization power of the evolved models: For the symbolic regression domain, the generalization power of the evolved expression was generally tied to the expression complexity in terms of its size, specifically it was widely assumed that reducing bloat would help reduce over-fitting. However, it has also been argued that bloat and over-fitting were two independent concepts based on evidence that not all GP systems with bloating over-fits and vice versa [152]. It has been further advocated that over-fitting might relate to functional complexity rather than the size of the expression [152, 110, 119]. The weighted complexity schemes we defined above aim to incorporate some indication of functional complexity based on the types of the individual operators.

5.8 G3P Search Space

Given a multi-dimensional dataset, it is possible to calculate the total number of 2D scatterplots that can be generated by G3P provided that the set of functional symbols and the maximum expression tree length are known.

Input: N dimensional (N variables) dataset $X = \{x_1, x_2, ..., x_N\}$

Output: 2D dataset $X' = \{x'_1, x'_2\}$ where $x'_1 = t_1(X)$ and $x'_2 = t_2(X)$ where $t_1(X), t_2(X)$ are transformation functions that are made up of the selected basis functions, coefficients and the variables of the input data.

Task: Evolve *transformation functions* t_1, t_2 using GP such that a fitness function F is optimized

Let the set of functional symbols be $\{+, -, *, /, sqr, sqrt, log, min, max\}$

Question: How many different 2D datasets are there if we limit the maximum tree depth to *d* ?

P=$\{x_1, x_2, ..., x_N, +, -, *, /, sqr, sqrt, log, min, max\}$

$P_0 = \{x_1, ..., x_N\}$

$P_1 = \{log, sqr, sqrt\}$

$P_2 = \{+, -, *, / min, max\}$

$a_{max} = 2$ (maximum arity)

n_d : total number of trees (transformation functions) with depth at most d

$$n_d = \sum_{a=0}^{a=a_{max}} |P_a| * (n_{d-1})^a, n_1 = |P_0| \tag{5.1}$$

As the recursive formula indicates, the expression trees are enumerated in a bottom-up manner. At first, all trees are single nodes (the operands) and the operators and more operands are added as the tree depth increases. Table 5.2 summarizes number of possible 1D transformation functions for each of the data mining problems we study throughout this thesis. The maximum tree depth is set to 17.[3]

[3]The reason for choosing maximum tree depth as 17 is historical. This is the default setting used in the ECJ toolkit based on Koza's original implementation. The tree depth can be set according to the number of variables in the dataset and also it does not need to be a static value.

Dataset	# variables	# possible 1D transformation functions
Accenting	45	$\approx 2^{529,842}$
BUPA	6	$\approx 2^{342,632}$
Crabs	5	$\approx 2^{326,129}$
Ionosphere	33	$\approx 2^{500,707}$
Olive Oil	8	$\approx 2^{368,898}$
PIMA	7	$\approx 2^{356,675}$
Segment	16	$\approx 2^{433,009}$
WBC	9	$\approx 2^{379,721}$
WDBC	30	$\approx 2^{491,766}$
Wine	13	$\approx 2^{413,709}$

Table 5.2: Total number of possible transformation functions for each dataset using G3P computed using equation 5.1 (maximum tree depth=17)

In order to generate 2D scatterplots, we have $(n_d * (n_d - 1))/2$ unique 1D transformation function pairings. Note that this is not the number of unique scatterplots. It is possible for a number of different pairs of transformation functions to generate the exact same visualization. The table shows that a brute-force search is out of the question even for a small dataset with a small number of variables and a moderate set of operators. Given the fact that it is computationally infeasible to enumerate and test all possible visualizations for any of these data mining problems, it is not possible to compare the results of one G3P run to the *ground truth*. Therefore, we compare our results to a number of standard dimensionality reduction methods such as the principle components analysis, multiple discriminants analysis and targeted projection pursuit.

5.9 G3P Experiments

In the preceding sections, we presented the G3P algorithm along with the various objective criteria. In this section, we present our experiments on the set of data classification problems that were introduced in section 2.5. The experiments are organized with respect to various aspects of the algorithm:

- *Compare the effect of various classifier assignment and classifiability computation policies for G3P*: we compare the effects of using multiple classifiers in one GP run with respect to the different aggregation schemes versus random classifier assignment schemes per G3P individual.

- *Compare results using G3P and a number of standard dimensionality reduction methods*: In

separate experiments, we apply the principal components analysis (PCA), multiple discriminants analysis (MDA) and the targeted projection pursuit (TPP) algorithms to project the original dataset into a 2D feature representation. For each new feature representation, we then apply a brute-force search for a classifier amongst the set of algorithms (Naive Bayes, K-nearest neighbors and the C4.5 decision tree) with the best predictive performance. We compare the standard methods to G3P in terms of the predictive power, the selected classifiers and the complexity of feature transformation functions.

- *Incorporate interpretability measures*: we incorporate the measures for the visual and semantic interpretability of the 2D features into the fitness function along with the classifiability criterion. We investigate three ways to compute a scalar fitness value as a weighted combination of the multiple objectives and population control based methods to prevent bloating with respect to the complexity of feature transformation expressions.

- *Compare the feature transformation functions and the visualizations generated by the G3P to the standard visualization techniques*: We present the scatterplots generated by the PCA, MDA and TPP and study each technique from the classifier performance, visual quality and feature complexity perspectives and make comparisons to solutions generated by G3P.

All experimental results are reported based on the 10x10-fold cross-validation scheme for a total of 100 runs on each dataset. All statistical significance tests are conducted using the corrected t-test as explained in chapter 4.

Algorithm 7 shows the evaluation of each GP individual and algorithm 8 shows the reporting of the best of run individual. This is the Standard GP implementation with a scalar fitness value where the multiple objectives are converted into one fitness value.

Algorithm 7: G3P Evaluate Method

```
Input : GPIndividual individual, Training dataset: dataset, folds: Number of Cross-Validation
        Folds
if NotEvaluated(individual) then
    if ContainsDuplicateExpressions(individual) then
        setAsBad(individual) ;
        exit();
    end
    output=computeMapping(dataset,individual.expressions);
    // If the mapping creates abnormal values such as NAN or
    //constant valued dimensions discard the individual
    if isBadMapping(output) OR ContainsConstantDimension(output) then
        setAsBad(individual) ;
        exit();
    end
    if hasClassifier(individual) then
        individual.classifiability= EvaluateClassifierCV(individual.classifier,output,folds) ;
        else
            //Individual needs a classifier assigned if it was created after initialization
            ComputeClassifiability(output); //Algorithm 6
        end
    end
    individual.visualInterpretability=ComputeVisualInterpretability(output);
    individual.semanticInterpretability=ComputeSemanticInterpretability(individual.expressions);
    individual.fitness=computeScalarFitness(individual);
end
```

Algorithm 8: G3P BestOfRun

```
//Compute K-fold cross validation accuracy on the transformed training set
mappedTrainingData=computeMapping(TrainingFold,individual.expressions);
trainingAccuracy=EvaluateClassifierCV(individual.classifier,mappedTrainingData,folds);

//Compute Accuracy on the test set
mappedTestData=computeMapping(TestFold,individual.expressions);
testingAccuracy=EvaluateClassifierCV(individual.classifier,mappedTrainingData,mappedTestData);
```

5.9.1 Comparing the Classifiability Policies

In this section, we investigate the effect of various classifier assignment and classifiability computation policies that were presented in section 5.4. As it was mentioned in that section, the most common approach has been to utilize one classifier per GP run and vary classifiers across different runs and compare the results. In each experiment, the evolved features will be biased towards the characteristics of the specific classifier that is utilized. For a set of N classifiers, this experimental set up requires N sets of GP experiments which can be computationally costly.

Sherrah et al. [135] and Smith [138] incorporate multiple classifiers within one GP run. This approach reduces the bias in feature construction by preventing the features to be tailored for one specific classifier. It also requires less number of GP runs for comparisons. In both methods, the number of features

that are constructed from the original dataset can vary as a part of the GP process. The authors report results based on random assignment of one classifier per GP individual. A natural question that arises is 'why not test all classifiers at once for each GP individual?'. Clearly, this increases the time needed to evaluate each individual, on the other hand, it might require a larger population/more generations for the same number of classifier evaluations where only one randomly assigned classifier is evaluated for each individual. The goal of the experiments in this section is to test this hypothesis. Specifically, we investigate if classifier performance aggregation has any advantage over random classifier assignment with respect to the classifiability of the evolved feature sets within one GP run. In this thesis, we do not further explore option of using one fixed classifier per run and repeating the runs for different classifiers due to computational cost and the feature construction bias.

There are a total of six different classifiability policies which are summarized in Algorithm 6. The G3P settings for this set of experiments are given in Table 5.3. Since, our goal is to compare the classifiability schemes, we left out the visual interpretability and expression complexity measures. All experiments were run using the same settings except for the classifiability policy. Three classifiers were utilized in these experiments: Naive Bayes, K-Nearest Neighbors (k=$\sqrt{\#\text{ of data items}}$) and C4.5 (J48) decision tree algorithm.

Each G3P run evolves 2D views of the training dataset using one of the six classifiability schemes and at the end of the run, the best individual is evaluated against the corresponding test set that was not used in evolving the data transformations during the run. Since there are 100 training set-test set pairs, each G3P run uses one pair and we report the results for each dataset based on those 100 runs.

We are interested in finding out if evaluating all classifiers at once would provide any advantage in finding good 2D feature representations for classification. Table 5.4 shows the classifier accuracy results on each dataset using the six classifiability policies. For each dataset, the accuracy is computed as the test set accuracy of the classifier associated with the best-of-run individual for each of the 100 runs.

We conducted a series of pairwise corrected t-tests for each dataset comparing the six policies (15

GP Type	Generational
Elitism	1 best G3P individual is preserved across generations
Population Size	100
Generations	50
Crossover Operator	Subtree Crossover
Crossover Probability	0.9
Reproduction Probability	0.05
Mutation Operator	Swap Mutate
Mutation Probability	0.05
Selection Algorithm	Tournament Selection
Tournament size	7
Functional Symbols	$\{+, -, *, protected/, min, max, power, log\}$
Ephemeral Random Constant (ERC)	$[-1, 1]$
Initialization	Ramped half-and-half (minimum depth:2, maximum depth:6)
Maximum tree depth	17
Classifiability Objective (C)	All classifiers min,max,mean,median, Random, Random&Parents
Visualization Objective (V)	T=2
Semantic Objective (S)	None
Evaluation	10 times 10-fold cross validation (total 100 runs)

Table 5.3: G3P Settings

pairwise tests in total with Bonferroni correction : $\alpha = 0.05/15$). For all individual datasets, no policy was found to be significantly different than others.

Dataset	All classifiers min	All classifiers max	All classifiers mean	All classifiers median	Random	Random&Parents	Original Best CV
Accenting	77.81 (3.25)	78.09 (3.05)	77.74 (3.03)	77.76 (3.19)	78.87 (3.08)	77.83 (3.26)	79.52 (3.05)
BUPA	68.02 (6.87)	68.41 (7.45)	68.70 (6.38)	69.51 (6.88)	68.08 (7.51)	68.09 (7.60)	65.68 (7.77)
Crabs	91.50 (7.12)	92.35 (6.94)	92.75 (5.43)	92.80 (7.05)	90.75 (7.99)	89.25 (9.57)	82.90 (8.94)
Ionosphere	88.86 (4.55)	88.60 (5.03)	89.17 (4.75)	88.69 (4.86)	87.41 (4.96)	88.38 (5.38)	90.66 (4.72)
Olive Oil	86.66 (3.97)	86.80 (3.60)	86.85 (3.96)	87.05 (4.09)	86.68 (3.38)	87.08 (4.00)	**95.34 (2.61)**
PIMA	75.10 (5.11)	75.57 (5.16)	76.01 (5.54)	75.69 (5.44)	75.60 (5.17)	75.39 (5.17)	75.23 (5.5)
Segment	87.07 (3.14)	89.53 (2.51)	88.53 (2.96)	88.18 (2.80)	88.89 (3.03)	88.23 (2.68)	**96.13 (1.49)**
WBC	96.28 (2.46)	96.09 (2.33)	96.19 (2.04)	96.31 (2.40)	96.27 (2.20)	95.91 (2.18)	97.26 (2.19)
WDBC	94.98 (3.08)	95.58 (2.88)	95.33 (2.98)	95.24 (3.02)	95.20 (2.69)	95.08 (2.97)	96.66 (2.21)
Wine	92.62 (6.68)	91.74 (6.58)	93.09 (5.77)	92.63 (5.57)	92.35 (6.60)	92.82 (6.09)	97.13 (4.10)
Overall	85.89 (10.09)	86.28 (10.03)	86.44 (9.86)	86.39 (9.84)	85.91 (10.07)	85.81 (10.18)	85.38 (11.92)

Table 5.4: Classifier accuracy (mean and standard deviation) results on 100 G3P runs using six classifiability policies. For each dataset, the accuracy is computed as the test set accuracy of the classifier associated with the best-of-run individual for each of the 100 runs.

Table 5.5 shows the running times per G3P run [4] on each dataset using each policy. The G3P utilizes a generational approach where parent individuals are replaced by the offsprings, we also incorporate elitism which serves as a global archive of best-of-run individual in order to ensure that the best G3P individuals found in one generation will not be lost throughout the run.

[4]We submit all GP runs for processing on a SGI cluster computing environment which executes around 50 runs in parallel. Therefore, the actual time it takes to complete 100 runs for each dataset is about two times as much as the time it takes to complete one run.

It is expected that run times for policies that use all classifiers for each G3P individual will be higher than the random assignment scheme. Since we utilized 3 classifiers, therefore, it is also expected that the runtime for the random&parents scheme will be similar to using all classifiers. Naturally, the difference in run times between the random schemes versus the schemes using all classifiers at once would increase as the number of classifiers increases.

Dataset	All classifiers min	All classifiers max	All classifiers mean	All classifiers median	Random	Random&Parents
Accenting	73.27 (13.43)	76.38 (6.82)	74.40 (11.92)	75.69 (9.67)	63.10 (8.63)	77.68 (14.42)
BUPA	2.98 (0.28)	2.97 (0.27)	3.03 (0.24)	3.00 (0.24)	2.10 (0.40)	3.12 (0.45)
Crabs	2.01 (0.13)	2.39 (0.44)	2.12 (0.16)	2.00 (0.12)	1.34 (0.08)	2.01 (0.19)
Ionosphere	2.78 (0.41)	3.03 (0.35)	2.86 (0.38)	2.89 (0.37)	1.96 (0.25)	3.10 (0.70)
Olive Oil	8.70 (0.91)	8.64 (0.83)	8.73 (0.96)	8.82 (0.67)	6.43 (0.50)	8.67 (1.29)
PIMA	6.29 (0.58)	6.75 (1.06)	6.74 (1.20)	6.11 (0.49)	4.35 (0.45)	6.09 (1.04)
Segment	48.26 (4.86)	46.71 (5.89)	46.85 (5.57)	43.40 (6.57)	36.25 (4.05)	44.43 (6.67)
WBC	8.62 (0.62)	8.18 (0.61)	7.98 (0.57)	8.19 (0.41)	6.01 (0.45)	7.62 (0.10)
WDBC	6.38 (0.55)	6.40 (0.60)	6.52 (0.48)	6.43 (0.54)	5.07 (0.46)	6.91 (0.98)
Wine	1.51 (0.19)	1.51 (0.18)	1.52 (0.19)	1.51 (0.18)	0.95 (0.10)	1.51 (0.19)

Table 5.5: Run-times in minutes per run using the six classifiability policies.

None of the policies that were using all 3 classifiers at once resulted in higher accuracy as opposed to the random policies at a statistically significant level. There is no statistically significant difference between the random and random&parents policies. However, as far as the computational burden is concerned, the random policy clearly stands out (table 5.5). Therefore, we conclude that the random policy can be safely preferred for all future experiments on these datasets based on empirical evidence that using all classifiers at once does not improve classification accuracy as opposed to a random classifier assignment scheme which is computationally the most efficient option.

We then compare the performance of the G3P (random policy) to the classifier performance on the original feature sets. The G3P algorithm searches for the best 2D feature set/classifier pair for each of the 100 training set-test set pairs that were created according to the 10x10-fold cross-validation scheme. In order to allow for a fair comparison, we utilize a similar evaluation scheme for the three classifiers on the original dataset. For each of the 100 cross-validation partitions of the dataset, we train all three classifiers on the training set using 10-fold cross validation and select the best classifier. Then, the selected classifier is evaluated on the corresponding test set and the result is recorded (algorithm 9). In

this scheme, there is no feature construction/dimensionality reduction, the classifiers take the original higher dimensional feature sets as input.

Algorithm 9: BestCV

```
foreach Cross-Validation Repetition (r=1 .. N) do
    foreach Training/Test set pair (f=1 .. P) do
        maxAccuracy=0;
        foreach classifier do
            trainingAccuracy=EvaluateClassifierCV(classifier,trainingData_{r,p},10-fold);
            if (trainingAccuracy > maxAccuracy)
            maxAccuracy=trainingAccuracy;
            bestClassifier_{r,p}=classifier;
        end
        //Compute accuracy of the best classifier on the test set
        testingAccuracy_{r,p}=EvaluateClassifierCV(bestClassifier_{r,p},trainingData_{r,p},testData_{r,p});
    end
end
```

The results are shown on the 'Original Best CV' column on table 5.4. On eight out of the ten datasets, the G3P is comparable to running classifiers on the full feature set (according to the corrected t-test). On these datasets, the advantage of G3P is that it enables data visualization without significantly sacrificing classification accuracy. However, for the Olive oils and Segments datasets, reducing the dimensionality came with a price of significant drop in classification accuracy. As far as the overall performance is considered, G3P with random classifiability policy is comparable to brute-force search for the best classifier performance with no dimensionality reduction.

5.9.2 Comparing G3P to Standard Dimensionality Reduction Methods

In this section, we compare G3P to predictive power of standard dimensionality reduction/classifier pairs. Principal Components Analysis (PCA), Multiple Discriminants Analysis (MDA) and Targeted Projection Pursuit (TPP) are applied to transform each dataset into a 2D representation. Algorithm 10 shows the BestCV algorithm with dimensionality reduction. The mapping function is induced from the training data and the training and test datasets are transformed using that mapping function before running the classifiers. The same three classifiers (Naive Bayes, K-Nearest Neighbors and C4.5 (J48) decision tree) are used here as well.

Algorithm 10: BestCV with Dimensionality Reduction

```
Input  : Method={PCA, MDA, TPP}
foreach Cross-Validation Repetition (r=1 .. N) do
    foreach Training/Test set pair (f=1 .. P) do
        maxAccuracy=0;
        M= LearnMapping(trainingData_{r,p}, Method);
        foreach classifier do
            trainingAccuracy=EvaluateClassifierCV(classifier,M(trainingData_{r,p}),10-fold);
            if (trainingAccuracy > maxAccuracy)
            maxAccuracy=trainingAccuracy;
            bestClassifier_{r,p}=classifier;
        end
        //Compute accuracy of the best classifier on the test set
        testingAccuracy_{r,p}=EvaluateClassifierCV(bestClassifier_{r,p},M(trainingData_{r,p}),M(testData_{r,p}));
    end
end
```

Table 5.6 shows the performance of the four dimensionality reduction/classifier combinations. The filter method selects two best data attributes using the Relief algorithm [81] prior to classification. The PCA, MDA and TPP methods project the dataset into 2D views using linear combinations of all data attributes. All four methods are compared to G3P using multiple pairwise corrected t-tests (at $\alpha = 0.05/6$). The results marked with an 'x' are significantly worse than the G3P. Only the MDA/classifier combination method performs significantly better than the G3P (marked with a ✓). On the Segment dataset, G3P outperforms all of the standard dimensionality reduction methods.

Dataset	Original Dimensionality	Filtered (2D) Best CV	PCA (2D) Best CV	MDA (2D) Best CV	TPP (2D) Best CV	G3P (Random) Best CV	I_{TS} Best PCA/MDA/TPP	I_{TS} Best G3P (Random)
Accenting	45	72.33 (3.03)	76.69 (3.08)	80.06 (2.82)	78.19 (3.51)	78.87 (3.08)	358	23
BUPA	6	62.19 (7.84)	56.58 (4.66) x	67.68 (7.49)	61.76 (6.72)	68.08 (7.51)	46	39
Crabs	5	56.70 (11.06) x	58.15 (10.27) x	93.65 (4.97)	72.95 (15.94) x	90.75 (7.99)	38	16
Ionosphere	33	83.61 (5.55)	80.09 (6.1) x	88.52 (4.81)	85.44 (10.45)	87.41 (4.96)	262	9
Olive Oil	8	86.17 (3.69)	85.61 (3.55)	82.68 (4.22)	72.96 (8.41) x	86.68 (3.38)	62	72
PIMA	7	75.61 (5.36)	74.47 (5.44)	77.73 (4.77)	73.72 (5.66)	75.60 (5.17)	54	40
Segment	16	85.49 (2.22)	72.34 (3.33) x	78.30 (2.50) x	77.86 (4.86) x	88.89 (3.03)	126	31
WBC	9	94.75 (2.41)	97.01 (2.01)	96.84 (1.96)	96.59 (2.06)	96.27 (2.20)	70	22
WDBC	30	93.64 (3.15)	93.04 (3.35)	96.49 (2.72)	92.29 (5.25)	95.20 (2.69)	238	33
Wine	13	85.00 (8.34) x	96.36 (4.00)	98.99 (2.30)✓	94.81 (4.85)	95.20 (2.69)	102	32
	Overall	79.55 (13.42)	79.03 (14.61)	86.09 (10.70)	80.66 (13.22)	85.91 (10.07)	135.6 (111.09)	31.7 (17.26)

Table 5.6: Results of standard dimensionality reduction techniques. 'x' indicates significantly worse and ✓indicates significantly better than G3P with random policy (see table 5.4). Also shown are total expression sizes for the best PCA/MDA/TPP versus the best G3P solution for each dataset.

Overall, the G3P performs at least as good as the standard methods with respect to discriminative power. However, G3P is computationally more expensive than the standard methods. This disadvantage can be counter-balanced to some extent by the fact that G3P can generate more compact and interpretable feature representations compared to the PCA, MDA and TPP. All three methods create

the two new features as weighted linear combinations of all features in the input dataset. As the number of features increases, the interpretability of the feature transformation functions decreases. For all three methods, a feature transformation function is in the form: $w_1 * v_1 + w_2 * v_2 + ... + w_D * v_D$ where w_i, v_i are the weights and values of the features and D is the dimensionality of the input dataset. Then, the total size of the feature transformation functions can be calculated as : $I_{TS} = 2 * (3 * D + D - 1)$. We note that, by inspecting the weights, it is possible to eliminate a number of data attributes and reduce the size of the expressions. However, this might also reduce the classifiability of the solution. Table 5.6 shows the total sizes of transformation functions corresponding to the best PCA/MDA/TPP versus the best G3P solutions. The results indicate that especially for higher dimensionality datasets, the reduction in expression size can be significant when G3P is used.

Note that the BestCV with dimensionality reduction algorithm is a pre-processing technique rather than a wrapper based feature extraction method. The feature transformations are computed independently from the classifier performance. Classifier performance serves as an assessment of the quality of the feature transformation, however the feature transformation method does not explicitly aim to increase performance of specific classifiers. The PCA method performs significantly worse than the MDA and TPP on the BUPA, Crabs, Ionosphere and Segment datasets. This is not surprising, PCA is not a supervised dimensionality reduction technique, therefore, it is not guaranteed to reveal class separation unless the directions of highest data variance are also the directions of class separation.

It appears that for some of these datasets, just selecting the two best original attributes using the filter method (the simplest transformation) works as good as more comprehensive 2D feature extraction algorithms (specifically on the PIMA and Olive oils datasets). Therefore, it would be wise to also consider these simplest transformations when choosing data models for the problem domain.

G3P is an adaptive technique which tries to find the best feature representation and classifier pair in an iterative manner where feature representations are evolved with an explicit objective of improving classifier performance and then selecting the best performing classifier. This joint approach separates G3P from the standard feature extraction techniques which either do not consider classifier performance

(filter) or focus on one classifier (wrapper and embedded) at a time.

5.9.3 Classifier Selection

In this section, we study how classifier selection is affected by the choice of dimensionality reduction technique. Figure 5.9 shows the percentage of times when each of the three classifiers were selected under each scenario we investigated above (the 'Best CV' algorithm with and without dimensionality reduction and the G3P algorithm with random classifier assignment policy).

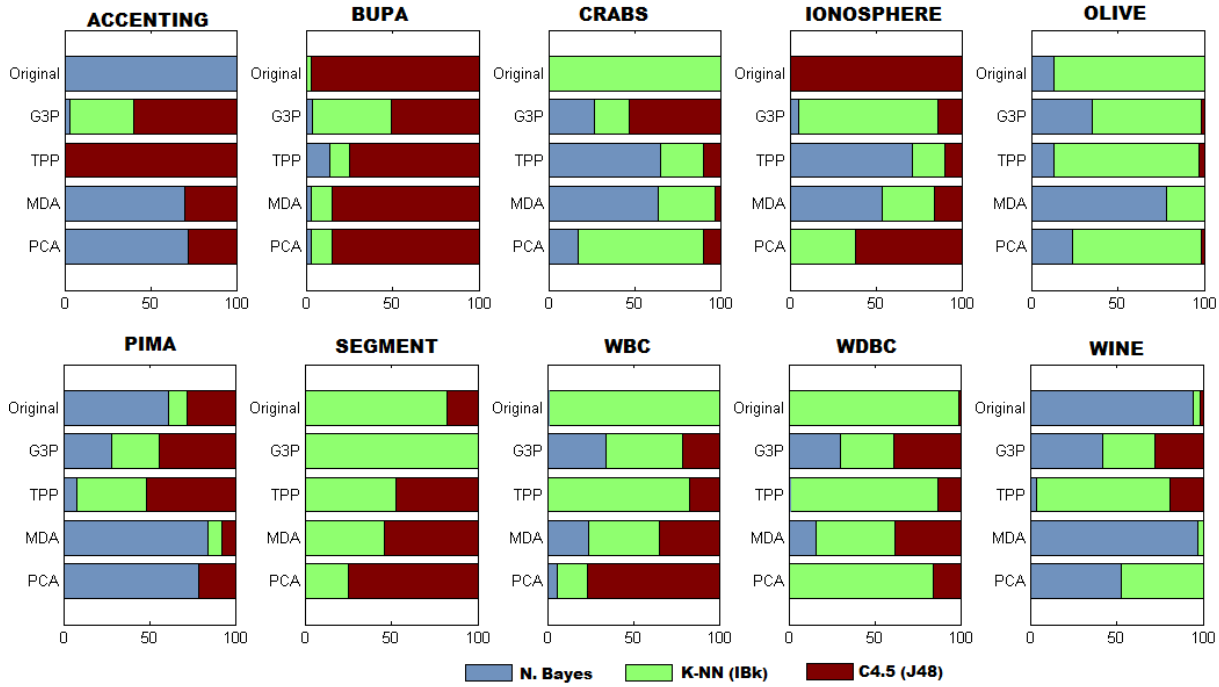


Figure 5.9: Selected classifiers without dimensionality reduction and using G3P, PCA, MDA and TPP methods

The case labeled with 'Original' identifies the runs without dimensionality reduction. We first consider the runs without dimensionality reduction. For almost all datasets, one classifier heavily dominates. The Naive Bayes classifier dominates for the Accenting, Wine and the PIMA datasets. The K-Nearest Neighbors classifier works best on the Crabs, Olive, Segment, WBC and the WDBC datasets. Finally, the C4.5 (J48) decision tree classifier outperforms the other two classifiers on the BUPA and the Ionosphere datasets. The results indicate that for each dataset, there is a classifier that matches the characteristics

of the dataset better than the other classifiers.

The PCA, MDA and the TPP results indicate that these dimensionality reduction techniques potentially change data characteristics to a point where another classifier might perform better. This effect is clearly visible on the Ionosphere dataset where the decision tree algorithm outperforms the Naive Bayes and the K-Nearest Neighbors classifiers on the original dataset. However, after dimensionality reduction using TPP and MDA, the Naive Bayes becomes the best performing classifier and the difference in performance is not significantly different than the performance without dimensionality reduction (see tables 5.4 and 5.6). Similarly, for the Crabs dataset, the K-Nearest Neighbors algorithm performs the best without dimensionality reduction, after MDA the Naive Bayes becomes the most frequent choice, indicating a change in data characteristics through data transformation. In this case, the MDA/classifier combination performs statistically significantly better (according to the corrected t-test) than classifying the original dataset without dimensionality reduction. This observation means that by changing the data characteristics, it is possible to find a better discriminative model using another data classification paradigm. Indeed, this is the motivation behind the feature extraction process.

It is also important to note that, the dimensionality reduction techniques demonstrate similar or dissimilar effects on different datasets. For example, the MDA and PCA show very similar classifier selection frequency patterns on the Accenting and BUPA datasets, however, they behave very differently on the Crabs and the Ionosphere datasets.

5.9.4 Incorporating Interpretability into G3P Fitness

In this section, we study the different ways to incorporate the interpretability measures into the G3P process in order to guide the evolutionary process towards interpretable feature representations. There are multiple ways to achieve this and we consider the following options:

- Direct manipulation of the fitness: devise a combination scheme on the multiple objectives for fitness computation.

- Manipulation of the selection operator: define priorities between the objectives where two individuals are compared with respect to the highest priority objective first and if they are equal, the second highest priority objective is used for comparison.
- Manipulation of the population: define a scheme where undesired individuals are detected and they are prevented from participating in the evolutionary process any further.

In the following sections, we divide the objectives into two groups: objectives related to discriminative power (classifiability and visual interpretability) and objectives related to the complexity of the feature transformation functions (semantic interpretability of the visualization axes). Within the context of this chapter, the measures of classifiability and visual interpretability are assumed to be equally important in computation of the discriminative power related fitness component.

5.9.4.1 Combination of Measures

Algorithm 11 shows the options we investigate. All options aim to balance the classifiability and visual interpretability of the extracted features while keeping the feature transformation functions as less complex as possible.

Algorithm 11: G3P ComputeScalarFitness

Input : GPIndividual individual
//All objectives are minimized (smaller values : better)
switch *fitness_option* **do**
case *CLASSIFIABILITY_INTERPRETABILITY*

$$\begin{aligned} fitness =& \alpha * individual.classificationError + \\ & \beta * individual.visualComplexity + \\ & \gamma * individual.individual.totalTreeSize \end{aligned}$$

case *SHERRAH_STYLE*

$$\begin{aligned} featureComplexity =& \gamma * uniqueAttributesCnt + \delta * individual.totalTreeSize \\ fitness =& \alpha * individial.classificationError + \\ & \beta * individual.visualComplexity + \\ & featureComplexity \end{aligned}$$

case *SMITH_STYLE*

$$\begin{aligned} featureComplexity =& \frac{(maxNodes - strength * individual.totalTreeSize - (1 - strength))}{(maxNodes - 1)} \\ fitness =& (\alpha * individual.classificationError + \\ & \beta * individual.visualComplexity) * featureComplexity \end{aligned}$$

endsw

Option 1: Weighted Combination

The first option (CLASSIFIABILITY_INTERPRETABILITY) is a linear combination of all three objectives:

$$fitness = \alpha * individual.classificationError + \beta * individual.visualComplexity + \gamma * individual.individual.totalTreeSize$$

where $\alpha + \beta = 1.0$. The visual complexity objective is computed using the visual quality measures, lower values indicate lower complexity, thus better views. The feature complexity objective can be computed using any of the semantic interpretability measures defined for the feature transformation functions. In this section, we consider the tree size as complexity for all experiments. Likewise, smaller values indicate less complex feature transformation expressions. $\alpha = \beta = 0.5$ assigns equal weights to the classifiability and visual interpretability objectives. The classification error is computed as a percentage, and the visual quality measures return values in the [0,1] range, therefore we multiply them by 100 in order to make their ranges comparable. The choice of the γ affects the trade-off between the phenotype based fitness versus the genotype based fitness. Too much emphasis on the feature complexity would bias the search towards less complex but potentially less useful feature representations. On the other hand, not enough emphasis of feature complexity encourages code growth (bloat).

Figures 5.10- 5.12 show G3P runs (with random classifier assignment policy) on the BUPA, PIMA and the Wine datasets. All plots are based on average of all 100 runs.

The left-most plots present the progress of the evolutionary process in terms of the accuracy of the best individual on each iteration. Each G3P experiment with a different classifiability-visual interpretability trade-off is displayed in a different color. The middle plots present the progress of the performance of the best individual on the test set[5]. The right-most plots show the average feature complexity (total tree size) at each generation. The 'classifier only' experiments utilize classifier accuracy as the fitness

[5]Note that the test set is not used during the G3P run in any way to drive the search, the test set performance is computed solely for the purpose of plotting the performance.

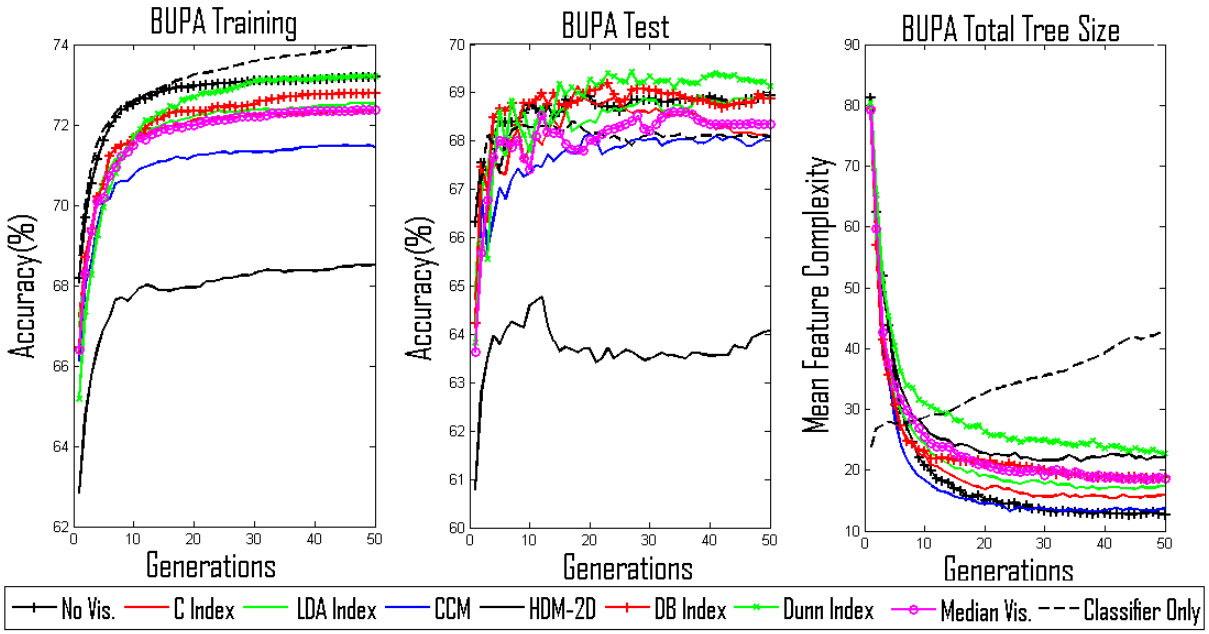


Figure 5.10: G3P results on the BUPA dataset with linear combination of the objectives. No Visualization: $\{\alpha$=1.0, β=0.0$\}$, otherwise: $\{\alpha$=0.5,β=0.5$\}$, feature complexity: Total Tree Size (I_{TS}), $\gamma = 0.1$

without any explicit constraint on the expression tree size (except for the default tree depth limit set to 17).

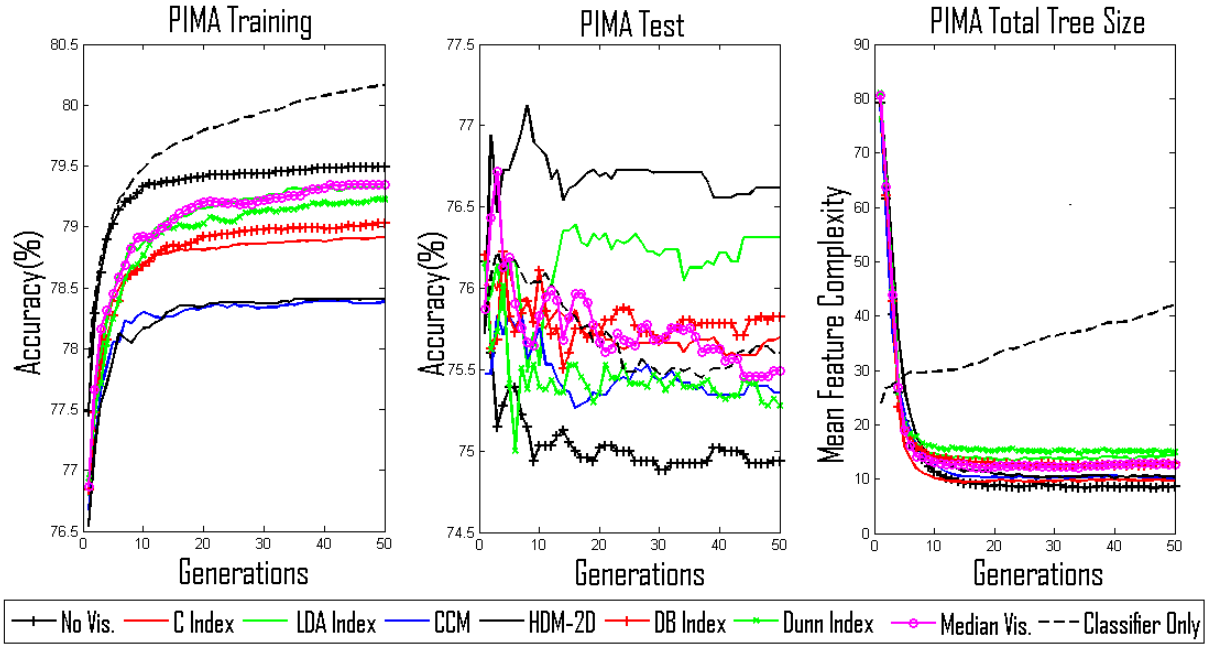


Figure 5.11: G3P results on the PIMA dataset with linear combination of the objectives. No Visualization: $\{\alpha$=1.0, β=0.0$\}$, otherwise: $\{\alpha$=0.5,β=0.5$\}$, feature complexity: Total Tree Size (I_{TS}), $\gamma = 0.1$.

The plots show that combining the feature complexity into the fitness effectively decreases the feature complexity while not reducing the classifiability (assessed on the test set) of the extracted features. Bloating is clearly evident on the 'Classifier Only' runs on the Wine dataset, where the average tree size keeps increasing while the fitness does not increase.

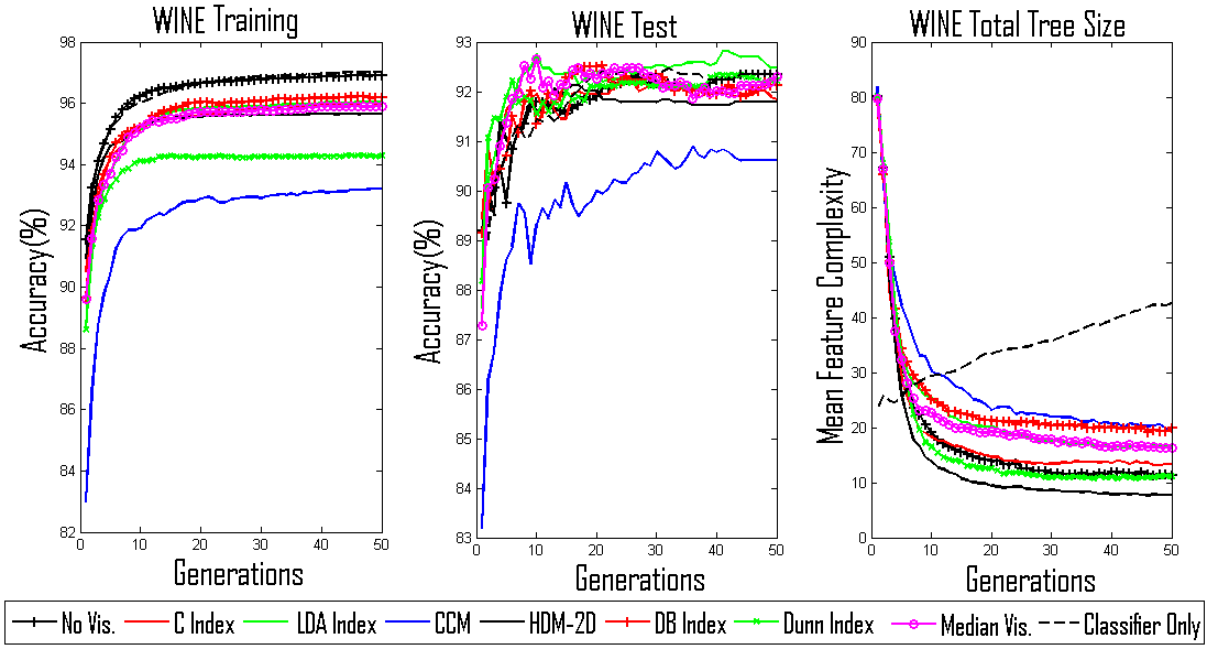


Figure 5.12: G3P results on the Wine dataset with linear combination of the objectives. No Visualization: $\{\alpha$=1.0, β=0.0$\}$, otherwise: $\{\alpha$=0.5,β=0.5$\}$, feature complexity: Total Tree Size (I_{TS}), $\gamma = 0.1$

Since there are multiple measures to assess visual interpretability, it is possible to use them individually or combine them into one measure. Here, we choose to take the median value of all six measures as the combined visual interpretability measure to assess each G3P individual and this aggregate measure are used for all of the following experiments.

Option 2: Sherrah Style Fitness

The Sherrah style fitness computation has been utilized by Sherrah et al. in [135] within the context of feature extraction using GP. Their measure considers classifier performance, the expression tree size and the number of attributes used in the feature transformation functions. The goal is to select those expressions that use the least amount of available features while keeping the size small and classification performance high. We modified this measure to include visual complexity measure as

well:

$$featureComplexity = \gamma * uniqueAttributesCnt + \delta * individual.totalTreeSize$$

$$fitness = \alpha * individial.classificationError +$$
$$\beta * individual.visualComplexity +$$
$$featureComplexity$$

where $\alpha + \beta = 1.0$. In [135], the values of γ and δ were set to 0.01 and 0.00001 respectively. The difference between this fitness computation and the previous one is the consideration of the number of the original data attributes that are included in the constructed features.

Option 3: Smith Style Fitness

Smith and Bull propose a fitness measure for GP based feature extraction in [139] which was based on the measure utilized in GP based classification rule mining by Bojarczuk et al in [24]. This measure balances the expression tree size and classification accuracy. We modified this measure to incorporate the visual complexity in conjunction with classifier accuracy:

$$featureComplexity = \frac{(maxNodes - strength * individual.totalTreeSize - (1 - strength))}{(maxNodes - 1)}$$

$$fitness = (\alpha * individual.classificationError +$$
$$\beta * individual.visualComplexity) * featureComplexity$$

where $\alpha + \beta = 1.0$. The strength parameter determines the importance of the tree size in determining the fitness. Unlike the first two methods of fitness computation, in this method, the feature complexity acts like a scaling factor on the phenotype fitness. Higher complexity increases the fitness (as smaller fitness means better individuals). In [139], the authors empirically found strength=0.04 to perform well across a number of data classification problems. Also, they set the value of maxNodes (maximum possible tree size) rather arbitrarily, in our experiments we set it to 2000.

Comparison of results:

Figures 5.13- 5.15 present the comparisons of the three methods that we utilized in fitness computations for G3P. In all experiments, the tree size complexity (I_{TS}) is utilized as the feature complexity measure and the median value of all six visual interpretability measures are utilized as the visual interpretability measure along with the random classifier assignment policy as the classifiability measure.

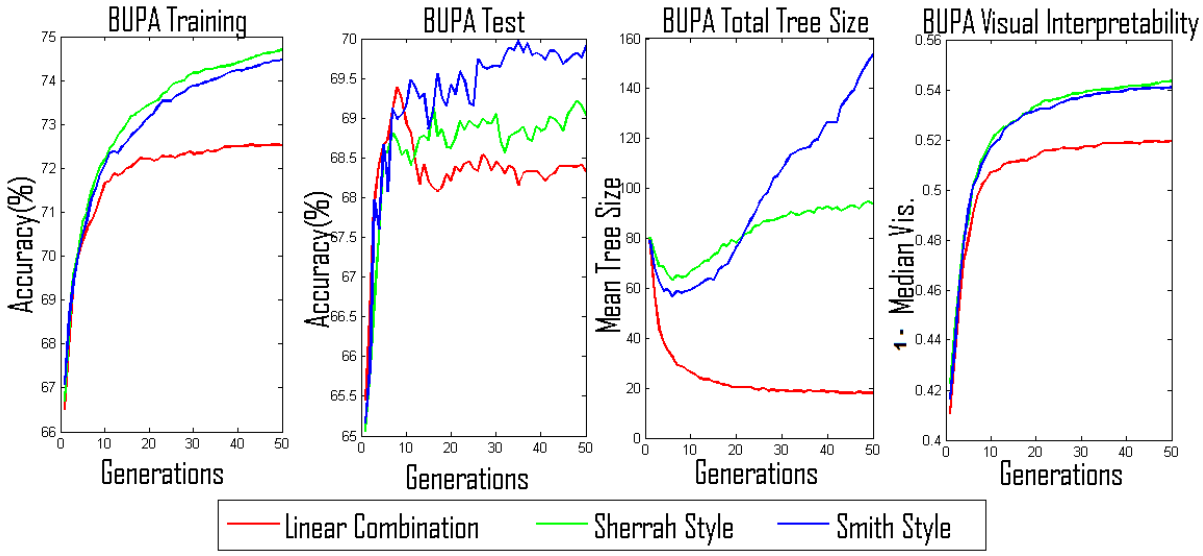


Figure 5.13: G3P results on the BUPA dataset with the three fitness computation methods

The training set, test set and visual interpretability plots show the average values across all 100 runs where higher values indicate better results. The feature complexity plot shows the average tree size at each generation across all 100 runs.

In all runs, the classifiability and the visual interpretability measures contribute to the fitness equally ($\alpha = \beta = 0.5$) while the contribution of the feature complexity is controlled through different parameters and values: $\gamma = 0.1$ for linear combination , $\gamma = 0.01, \delta = 0.00001$ for the Sherrah style and $strength = 0.04$ for the Smith style fitness computations.

For each of the datasets shown on the plots, the different choices of parameter values controlling the contribution of the feature complexity have the most effect on the average tree size throughout the evolutionary process. The linear combination method with $\gamma = 0.1$ has a trade-off that favors less complex individuals and the Sherrah and Smith style fitness computations with the values specified

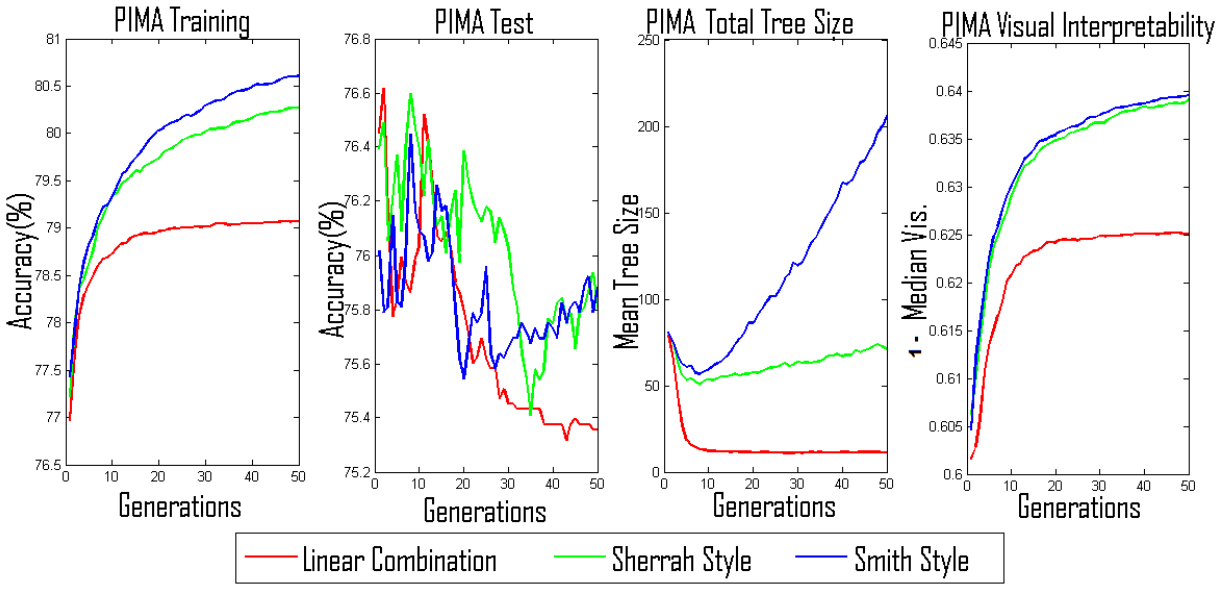


Figure 5.14: G3P results on the PIMA dataset with the three fitness computation methods

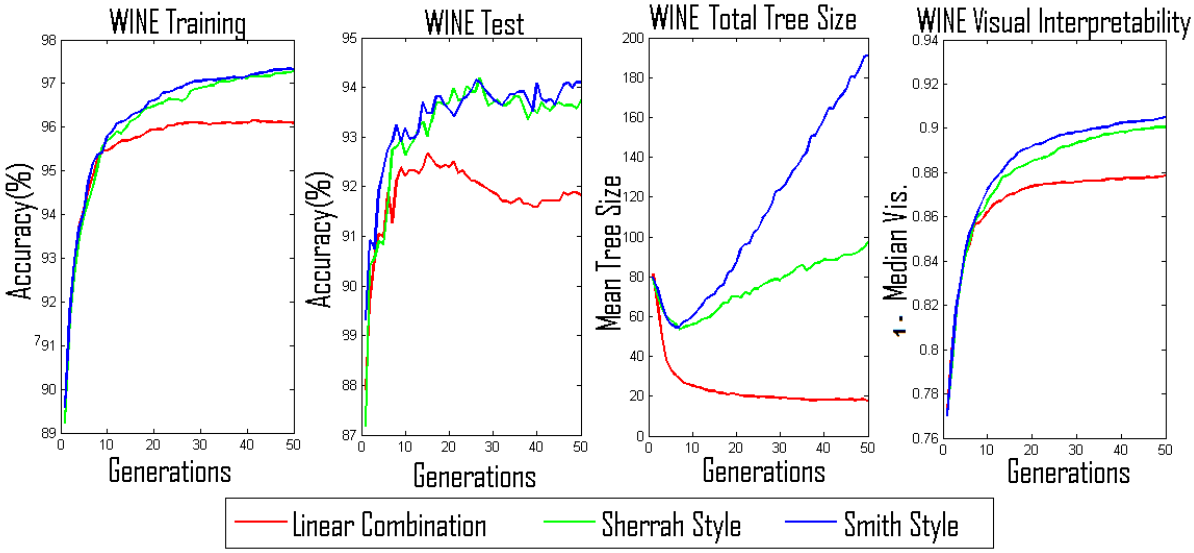


Figure 5.15: G3P results on the Wine dataset with the three fitness computation methods

above favor classifiability and visual interpretability over feature complexity. As a result, for all datasets shown here, the mean feature complexity increases as the generations progress when using the Smith style and Sherrah style fitness computations. The opposite effect is clearly visible when the linear combination method is used. We see that heavily favoring less complex feature transformations produce lower training set accuracy and visual interpretability at each generation. For the experiments on each dataset, average training set accuracy and visual interpretability of the best of runs are lower when

feature complexity is heavily reduced (figure 5.16(a,c)). However, when we look at the accuracy of the best of run individuals on the test sets across all 100 runs, we find that (corrected t-test α=0.05/3) there are no statistically significant differences amongst these methods (figure 5.16(b)). Based on the plots of each dataset, the Sherrah style fitness computation keeps the average tree size significantly lower than the Smith style fitness computation with the assigned parameters while keeping the training set performance and the visual interpretability comparable.

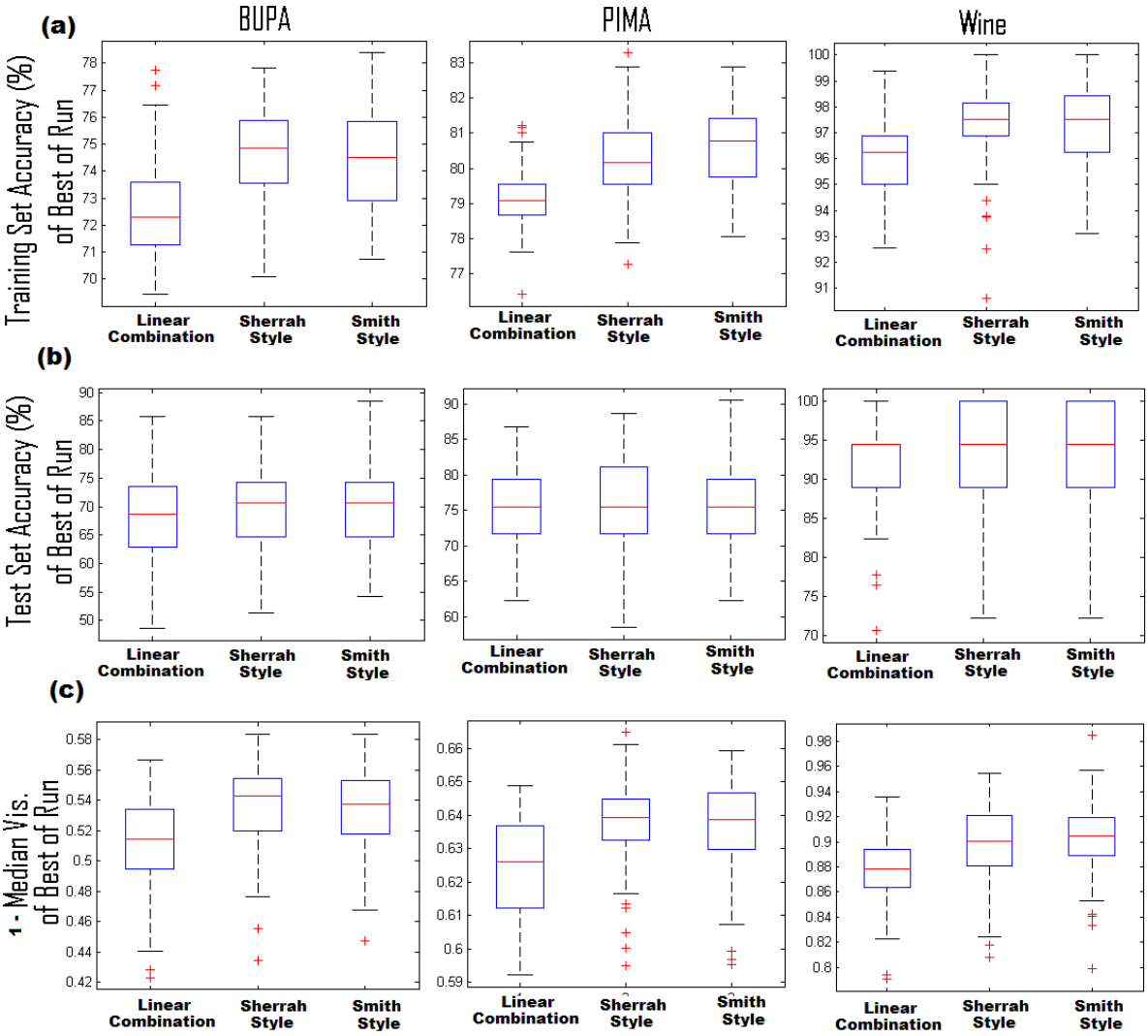


Figure 5.16: Comparisons of best of run individuals using the three fitness computation methods

5.9.4.2 Population Control Based Methods

Based on the results presented in the last section, we find that favoring less complex feature transformations effect the discriminative power. On the other hand, favoring discriminative power increases the feature complexity as the generations progress where a small increase in discriminative power comes with a cost of a significant increase in feature complexity. This is especially evident on the results where the Smith style computation is used. We conclude that finding an optimal accuracy-visual interpretability/feature complexity trade-off is computationally costly since the algorithm needs to be run multiple times with different coefficients that are present in each method of fitness computation.

In this section, investigate an alternative approach. Instead of incorporating the feature complexity directly into the fitness computation, we modify the selection operator to prioritize the objectives. As in previous section, we consider classifiability and the visual interpretability together as the discriminative power by taking the mean value of these two measures as the primary objective and the feature complexity as the secondary objective. In this scheme, the individuals are compared with respect to their discriminative power first, if they are equal than the individual with less feature complexity is preferred. We utilize the *Lexicographic Parsimony Pressure with Ratio Bucketing* method that was introduced in [98]. In this method, the individuals are compared based on their primary fitness value first, if they are equal then the smaller individual is considered better. The version with ratio bucketing employs a binning scheme in order to accommodate for continues fitness domains where the number of unique fitness values can be large. In this scheme, the lower fitness values are combined into larger buckets (coarse grain), while the higher fitness values separated into smaller buckets (fine grain). The size of the buckets are fixed with respect to a parameter r. Bottom $1/r$ of the individuals are placed into the lowest bucket and $1/r$ of the remaining individuals are placed into next bucket and the process continues until all individuals are placed into a bucket. We modify this method to consider any feature complexity method instead of just the expression tree size. In our experiments we set $r = 2$ as in [98].

However, prioritizing the objectives by itself does not prevent a highly complex feature transformation to be selected in the case that it offers even a slight increase in discriminative power. To counter-

balance this effect, we also incorporate a complexity control mechanism known as the *Tarpeian method* which was introduced in [116]. In this scheme, at each generation, before the breeding takes place, a portion of the individuals with above-average size are assigned the lowest possible fitness value in order to make it highly improbable for them to be selected. The proportion of the population that is to be demoted is controlled through a static parameter W. We modify this method to utilize any feature complexity measure instead of just expression size. In our experiments we set $W = 0.3$ as in [98].

We implement the two population control mechanisms in an incremental manner. First, we utilize the Lexicographic selection scheme alone and then along with the Tarpeian complexity control scheme. Figures 5.17- 5.19 present the comparisons of two schemes on the BUPA, PIMA and the Wine datasets to the three fitness computation methods from last section.

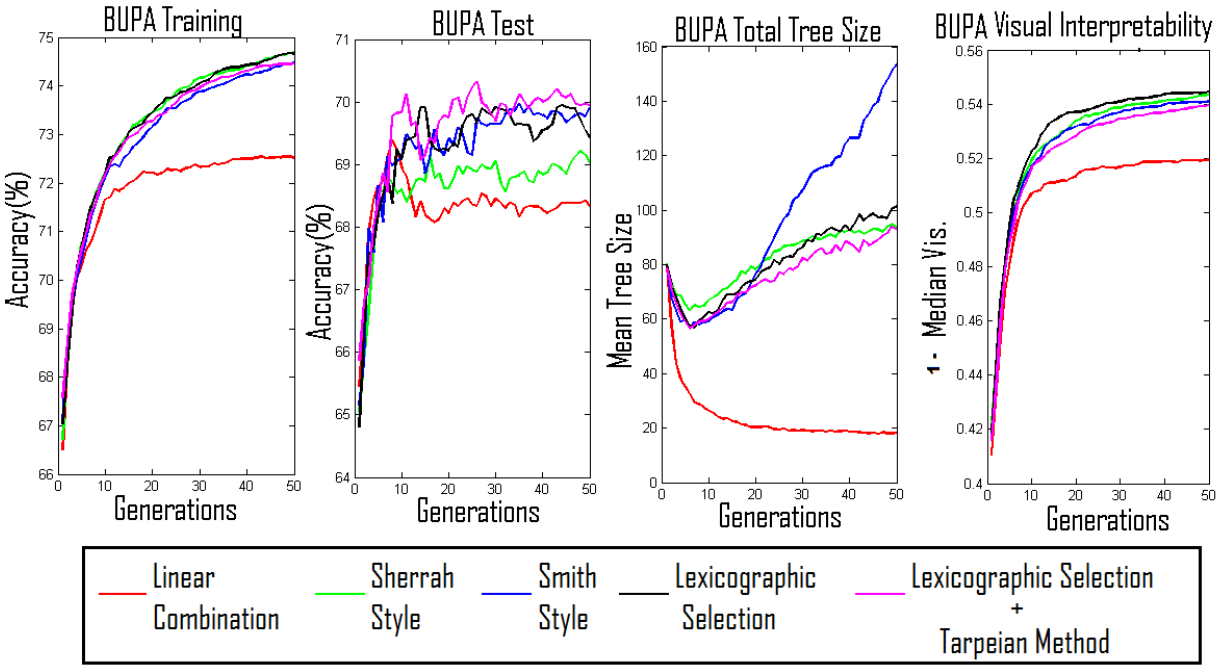


Figure 5.17: Comparison of the population control mechanisms to scalar fitness computation schemes

The results show that the both population control based methods are better on the average than the linear combination scheme in terms of the discriminative power and better than the Smith style fitness computation in terms of the feature complexity. Both methods behave similar to Sherrah style fitness

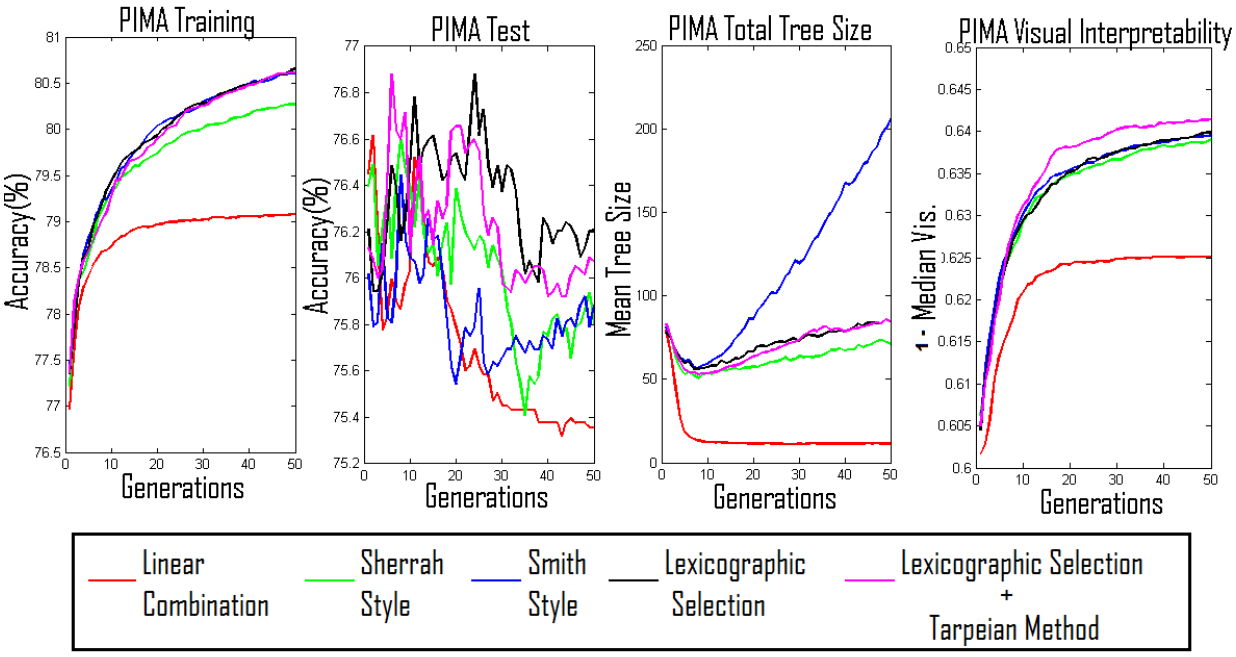


Figure 5.18: Comparison of the population control mechanisms to scalar fitness computation schemes on PIMA dataset

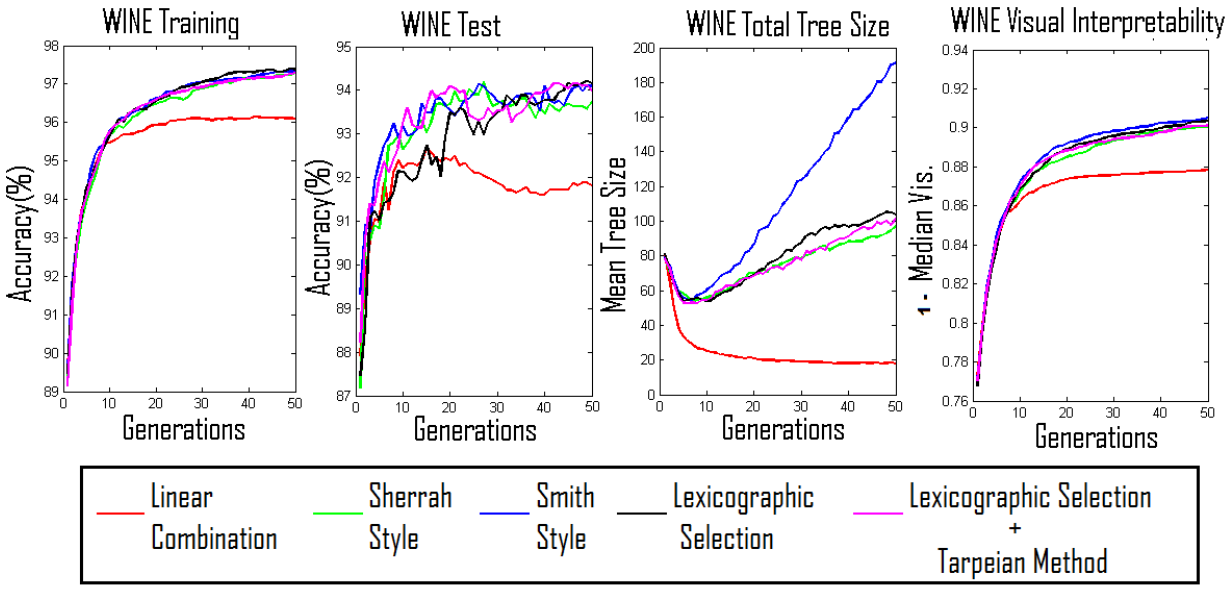


Figure 5.19: Comparison of the population control mechanisms to scalar fitness computation schemes on WINE dataset

computation. The results also indicate that combining the Lexicographic selection with the Tarpeian complexity control does not provide significant decrease in feature complexity in comparison to using the Lexicographic selection method alone.

5.9.4.3 Comparing Feature Complexity Measures

In this section we compare the experimental results on all datasets using the five feature complexity measures defined in section 5.6 to quantify the semantic interpretability of the visualization axes (feature complexity). All runs utilize the Lexicographic selection along with the Tarpeian complexity control scheme as in the previous section. All other G3P parameters are the same as given in table 5.3.

For each dataset, the results of separate experiments using each of the feature complexity measures are reported (tables 5.7- 5.16). The results on each dataset are presented with respect to four criteria: the training set and test set accuracy characterize the classification performance that can be reached on the feature representations generated by each corresponding G3P run, the median visual interpretability is the aggregated measure of all six visual complexity measures and the total nodes that are evaluated for each G3P run. All reported numbers are average values of the 100 G3P runs per separate experiment. Higher values of accuracy indicate better classifiability and lower median visual complexity indicates a better view. The total number of evaluated nodes can be taken as an indicator of computational burden on computing the transformed data if we assume that computational cost of different types of operators are same. Note that this is separate from the computational burden of computing the actual complexity value for each expression tree using different measures.

Since each feature complexity computation method returns different values for the expressions, their values are not directly comparable. Therefore, the total number evaluated nodes provides a common ground for comparing results. Based on analysis of the results on each dataset, we have not found any indication that using any of the five feature complexity measures caused a significant difference in terms of the four measures reported.

Specifically, the weighted schemes (WEC and WSC) that incorporate surrogate measures for operator linearity/nonlinearity did not improve classifiability (accuracy) nor did they lead to significant increase in total number of nodes evaluated given that they tend to rate larger expressions containing operators with smaller weights as less complex as opposed to shorter expressions containing operators with higher weights.

Measure	I_{TS}	I_{EC}	I_{WEC}	I_{SC}	I_{WSC}
Training Set Accuracy (%)	79.90 (1.05)	79.72 (1.17)	79.86 (1.02)	79.82 (1.15)	79.72 (1.06)
Test Set Accuracy (%)	78.64 (3.28)	78.10 (2.91)	78.24 (3.26)	78.22 (3.23)	78.17 (3.04)
Median Vis.	0.4 (0.016)	0.4 (0.016)	0.4 (0.013)	0.4 (0.016)	0.4 (0.014)
Total Nodes Evaluated	216,783 (95,764.3)	210,568 (85,116.4)	223,247 (95,520.2)	218,144 (106,221)	202,925 (99,798)

Table 5.7: G3P runs on the Accenting dataset.

Measure	I_{TS}	I_{EC}	I_{WEC}	I_{SC}	I_{WSC}
Training Set Accuracy (%)	74.45 (1.84)	74.72 (1.91)	74.54 (1.89)	74.72 (1.87)	74.51 (1.80)
Test Set Accuracy (%)	69.95 (7.10)	69.43 (6.90)	68.79 (6.79)	68.90 (6.88)	68.96(6.67)
Median Vis.	0.47 (0.026)	0.46 (0.027)	0.47 (0.024)	0.46 (0.024)	0.47 (0.027)
Total Nodes Evaluated	177,574 (81,350.2)	185,971 (104,273)	170,182 (83,942.3)	212,912 (99,582.8)	206,465 (107,894)

Table 5.8: G3P runs on the BUPA dataset.

Measure	I_{TS}	I_{EC}	I_{WEC}	I_{SC}	I_{WSC}
Training Set Accuracy (%)	95.67 (2.90)	95.94 (2.22)	96.22 (2,25)	96.3 (1.89)	96.17 (2.14)
Test Set Accuracy (%)	92.8 (5.83)	93.35 (5.99)	93.65 (4.92)	93.55 (5.47)	93.35 (5.08)
Median Vis.	0.11 (0.038)	0.11 (0.033)	0.10 (0.028)	0.11 (0.030)	0.11 (0.030)
Total Nodes Evaluated	226,187 (110,059)	249,884 (120,231)	230,233 (119,786)	254,268 (118,202)	249,961 (118,771)

Table 5.9: G3P runs on the Crabs dataset.

Measure	I_{TS}	I_{EC}	I_{WEC}	I_{SC}	I_{WSC}
Training Set Accuracy (%)	92.75 (1.33)	92.78 (1.43)	92.89 (1.29)	92.48 (1.49)	92.71 (1.44)
Test Set Accuracy (%)	90.20 (4.82)	89.45 (4.61)	89.95 (5.16)	90.77 (4.77)	90.00 (4.80)
Median Vis.	0.24 (0.04)	0.25 (0.05)	0.25 (0.05)	0.25 (0.04)	0.25 (0.05)
Total Nodes Evaluated	187,433 (95,617.7)	191,363 (107,744)	190,191 (98,125.6)	184,948 (102,190)	206,647 (121,131)

Table 5.10: G3P runs on the Ionosphere dataset.

Measure	I_{TS}	I_{EC}	I_{WEC}	I_{SC}	I_{WSC}
Training Set Accuracy (%)	87.39 (2.03)	87.61 (2.07)	87.58 (2.28)	87.63 (2.24)	87.33 (2.44)
Test Set Accuracy (%)	85.15 (4.55)	86.12 (3.73)	86.14 (4.17)	85.72 (3.65)	86.08 (4.19)
Median Vis.	0.14 (0.03)	0.14 (0.03)	0.14 (0.03)	0.14 (0.03)	0.14 (0.03)
Total Nodes Evaluated	146,016 (101,278)	137,600 (90,766.6)	132,726 (87,896.2)	143,460 (105,327)	143,282 (102,296)

Table 5.11: G3P runs on the Olive Oils dataset.

Measure	I_{TS}	I_{EC}	I_{WEC}	I_{SC}	I_{WSC}
Training Set Accuracy (%)	80.61 (0.99)	80.46 (1.10)	80.39 (0.99)	80.54 (1.12)	80.52 (0.92)
Test Set Accuracy (%)	76.07 (5.26)	76.24 (5.21)	76.31 (5.54)	76.49 (5.11)	76.32 (5.31)
Median Vis.	0.36 (0.01)	0.36 (0.01)	0.36 (0.11)	0.36 (0.01)	0.36 (0.011)
Total Nodes Evaluated	157,788 (83,729.5)	168,331 (101,531)	157,060 (92,655.1)	169,785 (95,832.1)	166,788 (97,281.1)

Table 5.12: G3P runs on the PIMA dataset.

Measure	I_{TS}	I_{EC}	I_{WEC}	I_{SC}	I_{WSC}
Training Set Accuracy (%)	89.43 (2.00)	89.42 (1.94)	89.17 (2.03)	89.57 (2.05)	89.55 (1.90)
Test Set Accuracy (%)	88.73 (2.91)	88.59 (2.6)	89.50 (3.23)	88.65 (2.91)	89.09 (2.79)
Median Vis.	0.17 (0.026)	0.17 (0.029)	0.17 (0.029)	0.17 (0.03)	0.16 (0.027)
Total Nodes Evaluated	122,259 (80,039.1)	113,434 (68,687)	114,678 (79,929.7)	116,788 (79,147.6)	118,235 (70,847.2)

Table 5.13: G3P runs on the Segment dataset.

Measure	I_{TS}	I_{EC}	I_{WEC}	I_{SC}	I_{WSC}
Training Set Accuracy (%)	97.80 (0.32)	97.82 (0.35)	97.83 (0.32)	97.83 (0.32)	97.86 (0.29)
Test Set Accuracy (%)	96.56 (2.16)	96.51 (2.05)	96.65 (2.26)	96.41 (2.29)	96.56 (2.37)
Median Vis.	0.08 (0.008)	0.08 (0.009)	0.08 (0.007)	0.08 (0.008)	0.08 (0.008)
Total Nodes Evaluated	276,530 (91,444.5)	301,392 (107,147)	300,046 (102,494)	300,745 (106,672)	286,746 (96,496.9)

Table 5.14: G3P runs on the WBC dataset.

Measure	I_{TS}	I_{EC}	I_{WEC}	I_{SC}	I_{WSC}
Training Set Accuracy (%)	96.90 (0.9)	96.97 (0.82)	96.84 (0.88)	96.88 (0.85)	96.97 (0.83)
Test Set Accuracy (%)	95.40 (2.84)	95.34 (2.98)	95.08 (2.86)	94.95 (3.07)	95.39 (2.96)
Median Vis.	0.16 (0.019)	0.16 (0.016)	0.16 (0.017)	0.16 (0.019)	0.16 (0.019)
Total Nodes Evaluated	231,245 (113,773)	230,528 (104,139)	220,390 (101,075)	228,027 (108,402)	221,754 (117,618)

Table 5.15: G3P runs on the WDBC dataset.

Measure	I_{TS}	I_{EC}	I_{WEC}	I_{SC}	I_{WSC}
Training Set Accuracy (%)	97.29 (1.43)	97.28 (1.41)	97.50 (1.44)	97.30 (1.42)	97.50 (1.49)
Test Set Accuracy (%)	93.99 (5.88)	93.86 (5.52)	94.29 (5.89)	94.54 (4.75)	93.48 (5.81)
Median Vis.	0.10 (0.028)	0.10 (0.029)	0.10 (0.028)	0.10 (0.028)	0.096 (0.027)
Total Nodes Evaluated	182,130 (101,443)	199,910 (116,819)	189,819 (106,464)	201,568 (119,801)	197,074 (101,971)

Table 5.16: G3P runs on the Wine dataset.

Intuitively, there is a correlation between the visual interpretability and classifiability. A closer look at the G3P results also confirms this. However, the degree of correlation may differ from dataset to dataset. Since accuracy varies between 0-100 and visual interpretability varies between 0-1, by looking at the results on each dataset, we can comment on relative quality of the evolved feature representations across datasets. For instance, the visual interpretability of the views are rated twice as good for the WBC dataset (0.08) than the views from the WDBC dataset (0.16) while the classification accuracy values on both datasets are very close.

5.9.5 Analysis G3P Visualizations and Comparison to PCA, MDA and TPP

In this section, we study the 2D feature representations generated for the visual classification of higher dimensional datasets using the G3P algorithm and make comparisons to three standard techniques commonly used in data mining. We first examine the visualizations and feature transformation functions generated by the PCA, MDA and TPP techniques and then examine the G3P results and present comparisons.

We perform dimensionality reduction/classification in order to jointly transform this dataset into a 2D representation and find a classifier with the maximum predictive power on this transformed dataset. A standard of way of achieving this goal was presented in algorithm 10 using the three dimensionality reduction methods: PCA, MDA and TPP and three classifiers: Naive Bayes, K-Nearest Neighbors (IBk) and the C4.5 (J48) decision tree classifiers. The algorithm searchers for the best data transformation/-

classifier pairs in a 10x10-fold cross-validation setting. Since there are a total 100 training set-test set pairs, the algorithm creates 100 pairs of data transformations and selected classifier pairs to choose from. The test set performance is important since it indicates how successful the data transformation/-classifier pair was on unseen data. On the other hand, the training set performance is also important since it indicates how well the algorithm learns from data. A high test set performance with a low training set performance is a suspect of having happened by chance. A low test set performance with a high training set performance indicates lack of generalization.

We compute the pareto-set for each set of runs using the PCA, MDA and TPP methods. In this section, we choose the 30-dimensional WDBC dataset (see section 2.5) as our case study to compare G3P to the standard techniques. Figure 5.20 shows the classifiability results for each experiment on the WDBC dataset along with the non-dominated solutions with respect to the training set-test set performance. In the plots, the classifier performance is shown as the classification error, therefore smaller values mean better performance and the pareto-set is expected to be closer to the origin.

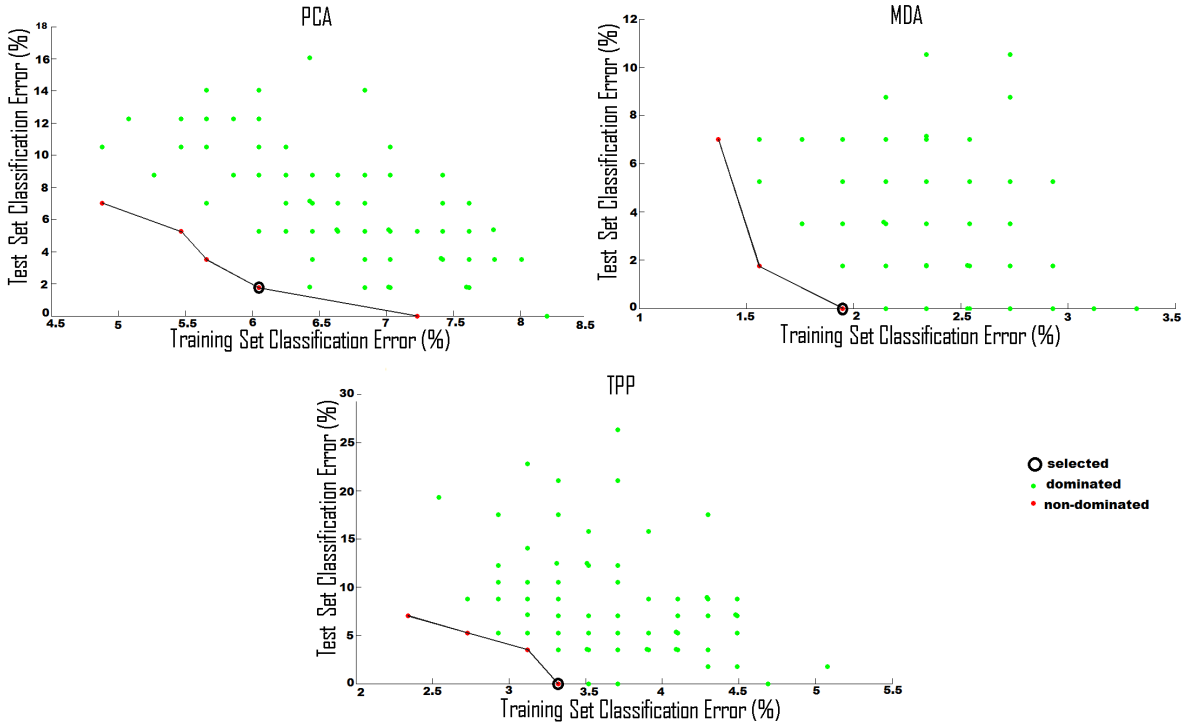


Figure 5.20: Selection of the best PCA, MDA and TPP visualizations amongst to 10x10-fold cross-validation runs

In each case, there are a small number of non-dominated solutions to choose from. Along the pareto-front of this solution set, favoring lower test set set error leads to a solution with a higher training set error. The choice is generally up to the decision maker. Here, we select one solution that is closest to the origin (0,0) with respect to the Euclidean distance: $argmin_i\{d_i = (tr_i * tr_i) + (ts_i * ts_i)\}$ where tr_i and ts_i are the training set and test set errors for each non-dominated solution i. The selected solutions according to this criterion are marked in figure 5.20.

Figure 5.21 shows the 2D scatterplots generated by the selected solution as explained above. For each visualization, the borders of each group are outlined.

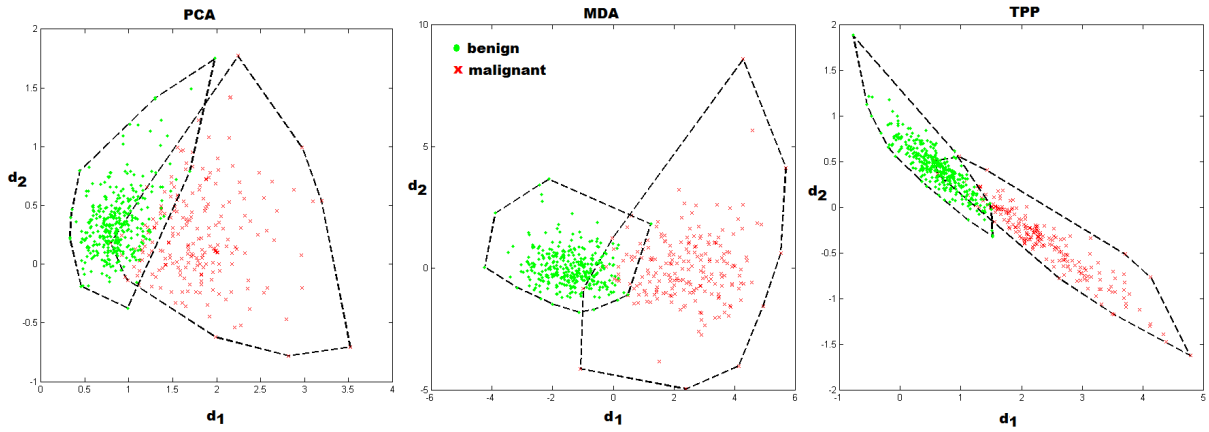


Figure 5.21: Selected PCA, MDA and TPP visualizations for the WDBC dataset

All three dimensionality reduction methods generate a linear combination of the original features as the visualization axes. The WDBC dataset contains 30 attributes all of which contribute to the two extracted features. As a result, the total size of the transformation function pair that maps the input dataset into a 2D visualization using any of these standard methods is large.

The following are the transformation functions that generate the 2D visualizations shown above using the PCA, MDA and the TPP methods:

PCA:

$$
\begin{aligned}
d_1 =& 0.264SDP + 0.261MP + 0.254MF + 0.243SDCP + 0.239WT + 0.235WCO + 0.234WR + \\
& 0.231WCP + 0.231WSY + 0.227MT + 0.225MR + 0.212MSM + 0.211SDSY + 0.205SDA + \\
& 0.203SDSM + 0.163MCT + 0.158WCT + 0.147SDCT + 0.134SDT + 0.128WP + 0.128WF + \\
& 0.125MSY + 0.117SDF + 0.111MCP + 0.11SDR + 0.094SDCO + 0.051MA + \\
& 0.032MCO + 0.013WSM + 0.013WA
\end{aligned}
$$

$$
\begin{aligned}
d_2 =& 0.347MA + 0.299WF + 0.253SDCO + 0.237MCT - 0.229MT - 0.228MR - 0.215WCP \\
& - 0.212WCO - 0.209WR + 0.202WP + 0.2SDT + 0.198SDCT - 0.192SDCP + 0.187MSY + \\
& 0.176MCO + 0.175SDSY + 0.171WT + 0.168SDF - 0.167SDSM + 0.152WSM - 0.125SDA + \\
& 0.124WSY - 0.107MSM + 0.106WCT + 0.074MP + 0.073WA - 0.054SDR \\
& - 0.032MCP - 0.025SDP + 0.011MF
\end{aligned}
$$

MDA:

$$
\begin{aligned}
d_1 =& -0.59MR + 0.052SDR + 0.026WR + 0.003MT - 12.219SDT - 13.67WT + 4.97MP + 14.42SDP + \\
& 1.54WP - 7.17MA + 0.75SDA + 0.13WA + 0.01MSM - 0.0006SDSM + 14.31WSM + 5.37MCT \\
& - 13.45SDCT + 9.11WCT + 17.08MCO - 8.66SDCO + 1.08WCO + 0.003MCP - 0.016SDCP \\
& - 0.0059WCP + 15.09MSY - 1.8SDSY + 2.14WSY + 4.88MF + 1.21SDF + 21.55WF
\end{aligned}
$$

$$
\begin{aligned}
d_2 =& 7.48MR + 0.029SDR - 1.84WR + 0.039MT + 8.12SDT + 66.94WT + 18.03MP - 18.37SDP + \\
& 2.16WP - 56.23MA - 6.97SDA + 0.27WA - 0.44MSM + 0.05SDSM + 15.47WSM - 7.47MCT \\
& - 20.40SDCT + 64.88WCT + 17.63MCO + 37.47SDCO0 + 1.88WCO - 0.04MCP + 0.18SDCP \\
& - 0.02WCP - 16.32MSY - 4.06SDSY + 0.78WSY - 7.73MF - 4.94SDF - 1.58WF
\end{aligned}
$$

TPP:

$$
\begin{aligned}
d_1 =& -0.06MR + 0.24SDR - 0.02WR + 0.11MT - 0.59SDT + 0.11WT + 0.74MP + 1.24SDP \\
& - 0.15WP - 0.81MA + 0.46SDA - 0.13WA + 0.28MSM + 0.15SDSM - 0.11WSM - 0.47MCT \\
& - 0.29SDCT - 0.46WCT - 0.2MCO - 0.02SDCO + 0.5WCO + 0.57MCP + 0.42SDCP + 0.39WCP + \\
& 0.3MSY + 0.33SDSY + 0.54WSY + 1.05MF + 0.32SDF + 0.24WF
\end{aligned}
$$

$$
\begin{aligned}
d_2 =& 0.43MR + 0.06SDR + 0.35WR + 0.03MT + 0.73SDT - 0.22WT - 0.72MP - 1.03SDP + \\
& 0.26WP + 0.7MA - 0.34SDA + 0.21WA - 0.25MSM - 0.21SDSM + 0.27WSM + 0.33MCT \\
& + 0.21SDCT + 0.49WCT + 0.2MCO - 0.09SDCO - 0.11WCO - 0.17MCP - 0.11SDCP - 0.28WCP + \\
& 0.07MSY - 0.26SDSY - 0.37WSY - 0.58MF + 0.01SDF - 0.11WF
\end{aligned}
$$

Now, we look at how the visualizations generated by these three standard methods are rated by the visual quality measures. Figure 5.22 shows the boxplots for each measure across all 100 runs using the PCA, MDA and the TPP methods. For each measure, smaller values indicate better views according to that measure. Across all individual measures, the views generated by MDA or TPP are

rated as significantly better than the views generated by the PCA. The views generated by MDA are better than those generated by TPP based on the C Index, CCM and the HDM-2D measures and the views generated by TPP better than those generated by MDA based on the LDA Index, DB Index and Dunn Index.

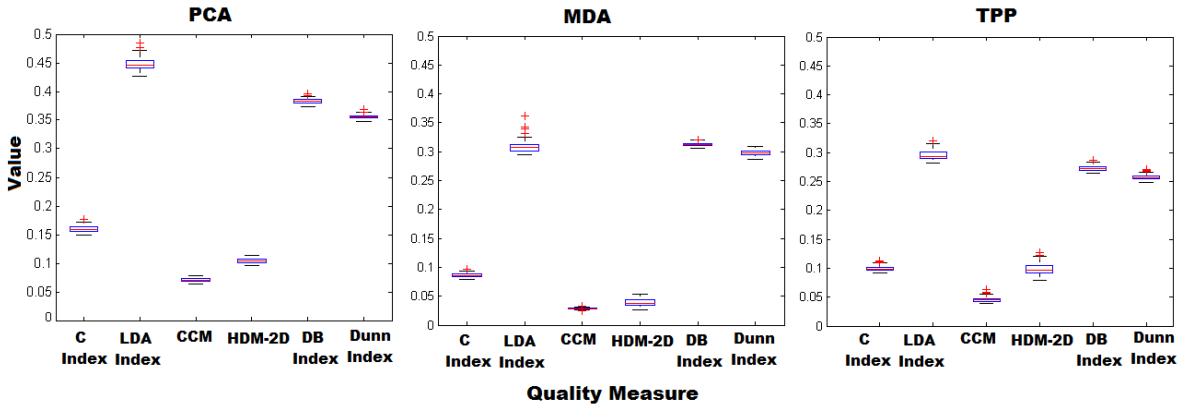


Figure 5.22: Visual Quality Values on the 2D visualizations generated by the PCA, MDA and the TPP algorithms. Smaller values mean better views.

Figure 5.23 shows the comparisons of each method based on the classification error on the training set, test set and the median value of all six visual interpretability measures.In terms of the classifiability criterion, the MDA performs significantly better (according to pairwise corrected t-test ($\alpha = 0.05/3$). However, in terms of the median visual interpretability, the views generated by the TPP method are rated significantly better than those by the MDA and PCA methods.

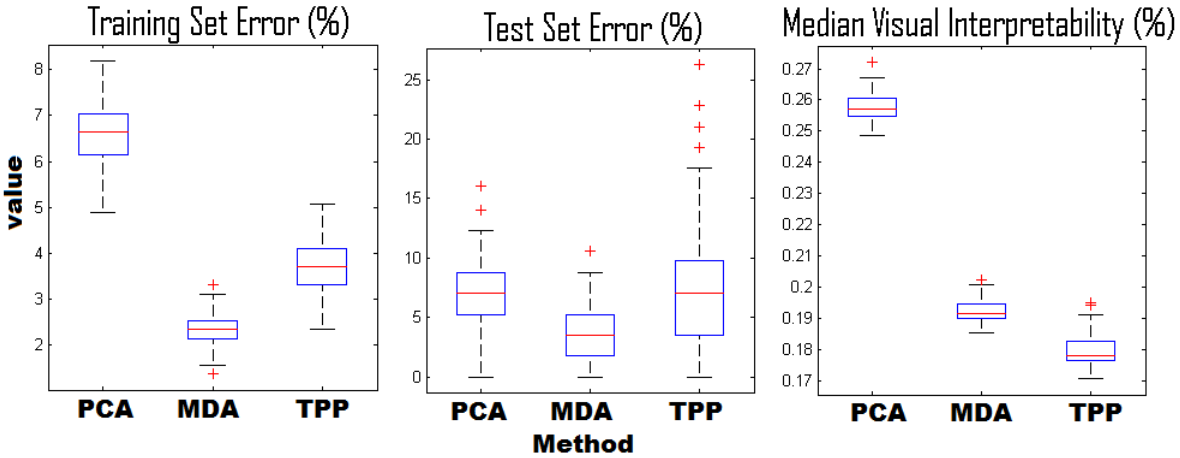


Figure 5.23: Comparisons of training set error, test set error and the median visual interpretability of the views generated by PCA, MDA and TPP. For all measures small values indicate better results.

There is no significant difference between the MDA and TPP in terms of generalization power (test set error) indicating that both methods are equally good for creating feature representations for visual classification while the TPP method generates views with higher visual interpretability in terms of an aggregate of the six measures. This observation reinforces the idea of searching for feature representations that are both classifiable in terms of classification performance and better views of the data in terms of the visual interpretability measures.

The results of the G3P runs are shown in figure 5.24. The most significant difference is in terms of the sizes of the feature transformation functions.

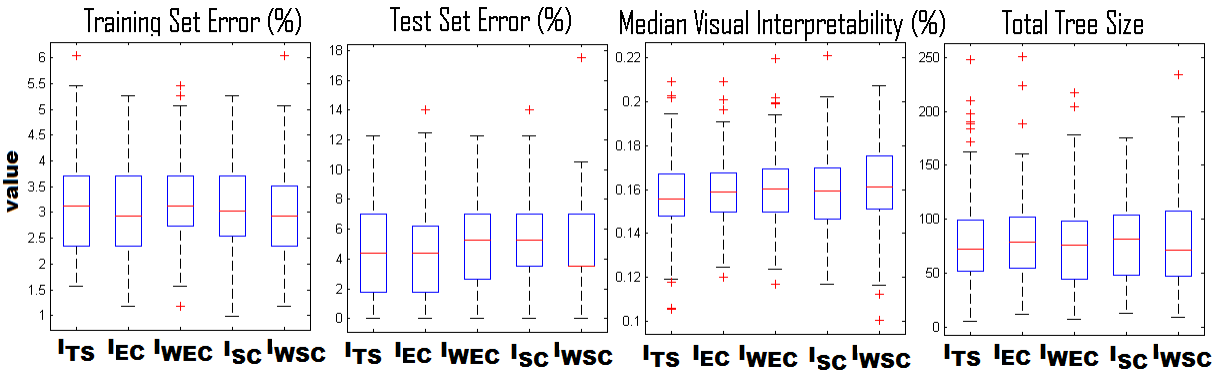


Figure 5.24: G3P results on the WDBC dataset

As shown above, the transformation functions for PCA, MDA and TPP are of fixed form and size ($I_{TS} = 238$). G3P generates feature transformation functions which are much smaller while keeping the discriminative power comparable to these methods. In terms of feature complexity, the G3P provides a more diverse set of options for the users to choose from. Figure 5.25 shows a possible scenario of choosing a solution amongst the G3P results with the I_{TS} as the feature complexity criterion.

We explore the solutions from the perspective of two logical groups. The training set and test set error rates are grouped into the classification performance and the visual and feature complexity criteria are grouped into the interpretability characteristics of G3P results. The solutions colored in red are the members of the pareto set on each plot. A number of selected solutions and their values on the training set-test set and visual interpretability-feature complexity trade-offs are shown on each plot. According to the results, the two best solutions according to classifiability criterion are not amongst the best solutions

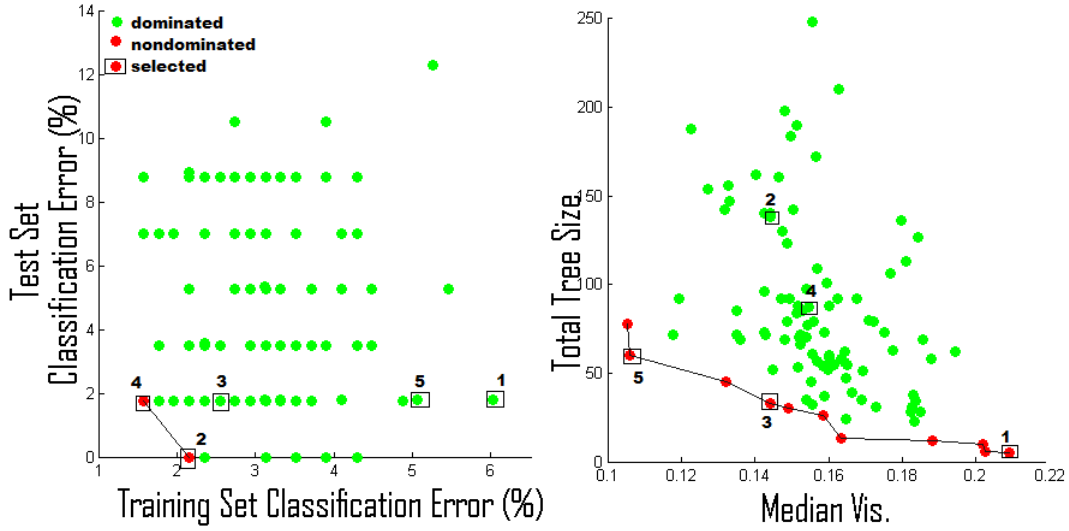


Figure 5.25: Training-Test set classification error and Median Vis.-Feature Complexity trade-offs in G3P results on the WDBC dataset

with respect to the visual and semantic interpretability of the generated feature representations. If the user wants to favor classifiability over the interpretability criteria, these two solutions can be considered. Alternatively, if the user favors interpretability, the solutions on the interpretability pareto set can be explored. Amongst the solutions shown in these plots, we choose the solutions marked as '1' and '3' to further explore here, because of their low feature complexity.

Figure 5.26 shows the expression trees corresponding to the pair of feature transformation functions for the selected G3P solution '3'. The total size of the expression trees is 33 as opposed to 238 of the PCA, MDA and TPP methods. The classifier that is associated with this solution is the K-Nearest Neighbors (IBk) algorithm.

As it is shown on the expression trees, there are a number of easily identifiable compact sub-expressions. Identifying these sub-expressions help simplify the feature transformation functions and also each of these sub-expressions may correspond to a new and meaningful combinations of a subset of the attributes for the researchers to consider. The expressions for the selected G3P solutions can be organized as follows:

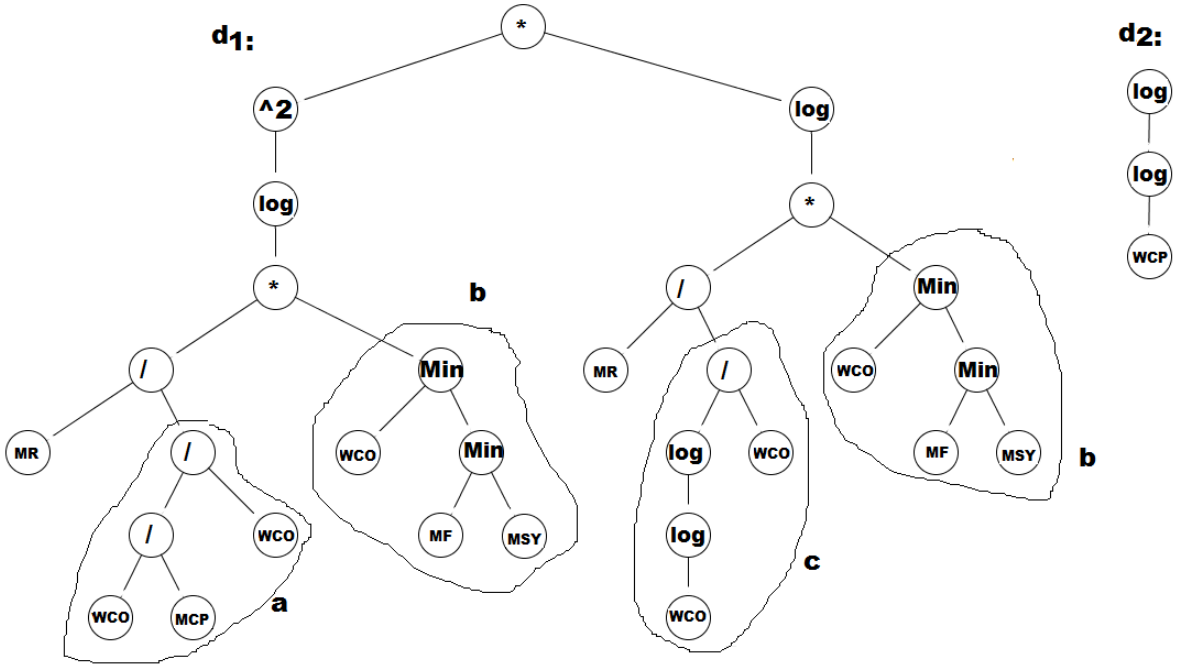


Figure 5.26: Feature transformation expressions for the G3P solution '3' on WDBC dataset

$$d_1 = (log(\frac{MR}{a} * b))^2 * log(\frac{MR}{c} * b)$$

$$a = \frac{1}{MCP}$$

$$b = Min(WCO, MF, MSY)$$

$$c = \frac{log(log(WCO))}{WCO}$$

$$d_2 = log(log(WCP))$$

This set of feature transformation functions utilize only 6 of the original 30 attributes in the dataset indicating that 80% of the original attributes are not needed if this solution is selected for analysis of the WDBC dataset.

The expression trees generated by G3P solution '1' are given in figure 5.27. These feature transformation functions are very simple: $d_1 = log(SDCP)$ and $d_2 = log(log(MCP))$. In this solution, only 2 out of the 30 original attributes are needed. The selected classifier for this solution is the Naive Bayes algorithm.

The visualizations generated by these two G3P solutions are presented in figure 5.28. Solution

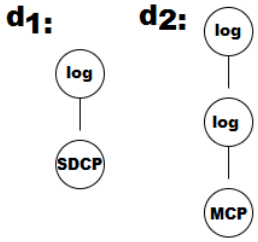


Figure 5.27: Feature transformation expressions for the G3P solution '1' on WDBC dataset

'1' offers very simple visualization axes, while the visualization generated by solution '3' contains less overlapping points between the groups thus indicating better separation.

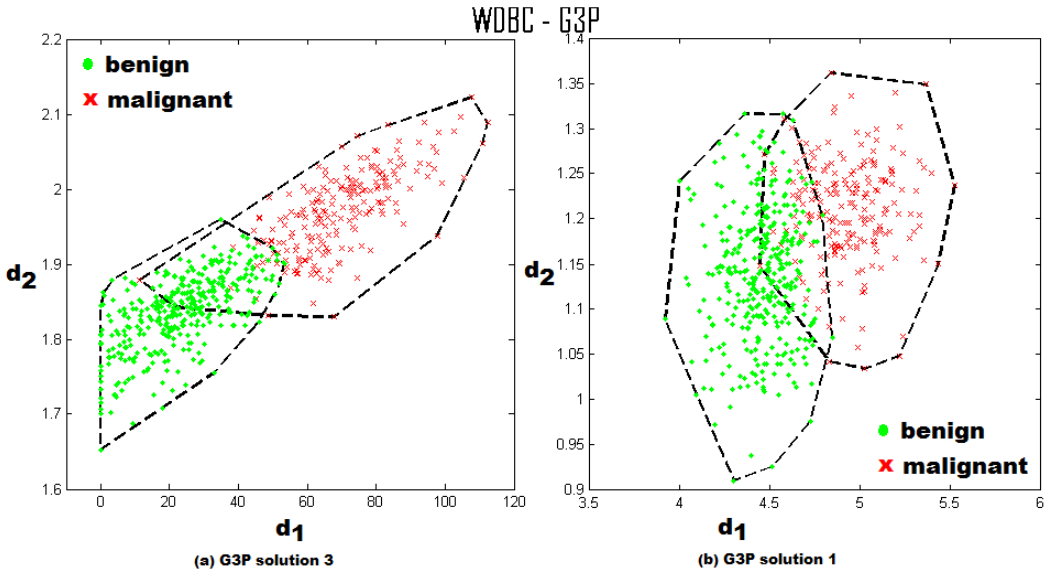


Figure 5.28: 2D scatterplot of the WDBC dataset generated by the selected G3P solution

The evolutionary process of G3P is not interactive; the algorithm runs without user intervention to guide the evolutionary process. Once this process is completed, the rich set of generated solutions can be manually explored from different perspectives corresponding to the classifiability and interpretability criteria.

5.10 Discussion

In summary, the results indicate that G3P can compete with the standard dimensionality reduction/-classifier combination methods in terms of predictive power. Though the algorithm is significantly more

computationally expensive as opposed to its counter-parts, we show that the difference is well compensated by the fact that the G3P can generate multiple feature representations that are more compact and interpretable which is an advantage for exploratory analysis of higher dimensional datasets. Moreover, none of the standard techniques explicitly consider multiple objectives such as the classifiability, interpretability of the visualizations and the visualization axes simultaneously as the G3P does.

In this section, we considered a fixed trade-off between the classification performance and the visual interpretability of the visualizations by linearly combining them as a part of the fitness measure with equal weights which is equal to taking the average of the two measures (after scaling the visual interpretability values to same range as the accuracy values). However, different weighting schemes might have given us different results. The user can favor visual interpretability over classification accuracy or vice versa by assigning larger weight for the favored objective before the G3P run.

The G3P is an automatic exploratory data analysis tool where the evolutionary process automatically creates a number of 'optimal' feature representations and classifier pairings with respect to predictive performance and interpretability. Therefore, the user is spared from manually exploring the vast space of possible scatterplots looking for *interesting* structures in the classical exploratory data analysis sense. However, the G3P still leaves space for manual exploration by providing the users with multiple solutions which are better than others from different perspectives. We have shown an example scenario on the 30-dimensional WDBC dataset in section 5.9.4.3. The user can either favor classification performance over interpretability or vice versa. In each case, the choice leads to a different feature representation that can be further explored and contrasted with other options. Although a similar exploration can also be done using the standard techniques, the G3P provides a more diverse set of solutions to explore.

Chapter 6

Multi-Objective Genetic Programming Projection Pursuit (MOG3P)

The G3P algorithm is by definition a multi-objective method and it can be implemented in a number of ways. In the last chapter, we explored the following options:

- *Weighted combination of the objectives*: We explored three ways to compute a scalar fitness value based on the classifiability, visual interpretability and feature complexity objectives.
- *Prioritizing the objectives*: We unified the classifiability and visual interpretability measures into one measure through a weighted linear combination that characterized the discriminative power of the G3P solutions which was separate from the feature complexity. We then explored the efficacy of a lexicographic selection operator which compared the G3P solutions based on their discriminative power first and in the case of equivalence, solutions with lower complexity were favored.
- *Prioritizing the objectives along with explicit feature complexity control*: We combined the objective prioritization scheme described above with an explicit population control mechanism which demoted the individuals with above average complexity with respect to a preset probability.

In this chapter, we first discuss the shortcomings of these approaches and then introduce a Pareto based multi-objective optimization scheme for G3P which is called the Multi-objective Genetic Programming Projection Pursuit (MOG3P). Various MOG3P experiments along with comparisons to G3P are presented. We then introduce a seeding based technique to incorporate best known feature transformation functions generated by the standard dimensionality techniques into the MOG3P as a hybrid approach. Finally, we study the symptoms of over-fitting in MOG3P runs and present a validation method to control over-fitting during the evolutionary process.

6.1 Related Work

Machine learning problems are inherently multi-objective. The most re-occurring example is the accuracy versus the complexity of the learning algorithms. For instance, different decision tree classifiers on the same dataset are evaluated in terms of their accuracy versus the number of nodes they contain or the number of unique attributes they need to model the data. Various simplification schemes have been proposed to simplify the trees by pruning redundant branches thus reducing the complexity of the classifier (see for example: [66]). A similar case applies to the feature extraction problem where the utility of the set of extracted features versus the number and complexity of extracted features provide the researchers with multiple options to consider. Freitas identifies three approaches for multi-objective machine learning and presents a discussion of each method along with the arguments in favor of or against each case: [56]:

- Transforming the multi-objective problem into a single objective problem via a weighted formula
- Prioritizing the objectives
- Pareto approach

The arguments presented by the author apply to the general machine learning research without targeting a specific implementation framework such as evolutionary computing. The author concludes that

even though it is by far the most common approach in the machine learning field, the weighted formula approach is an ad-hoc technique where the prioritization and Pareto approaches are more principled methods and should receive more attention from the machine learning and data mining communities.

Machine learning and data mining problems have attracted interest from the evolutionary computing community as well. For example, symbolic regression has been one of the earliest applications of genetic programming that was introduced by Koza [83]. Various multi-objective optimization schemes have been proposed for a number of data analysis tasks including general purpose data classification [19], feature extraction [163, 164, 138], data visualization [151, 150] and specific data mining applications such as in medicine [162] and direct marketing [21]. Detailed overviews and examples of various multi-objective machine learning applications including a number of evolutionary computing techniques can be found in collections [73, 60, 112].

In this section, we mainly concentrate on the closest work to ours with respect to the representation where tree based genetic programming is used with tree size (complexity) an explicit objective and the problem domains related to general purpose data classification and feature extraction including data visualization.

Bleuler et al. advocates the use of tree size as an explicit objective besides the fitness value based on functionality of the evolved expressions within the SPEA2 framework in [23]. The authors report that this multi-objective scheme is successful in reducing bloat and helping GP evolve compact programs which solve a benchmark problem that is known as the 'k-parity'. Bernstein et al. propose a multi-objective GP technique [19] that evolves decision tree like data classifiers with accuracy and the tree size as the two objectives. The algorithm is called 'Pseudo-Objective Parsimony Enforcement GP' (POPE-GP) and the implementation is based on the NSGA-II optimization framework. Similarly, Parrott et al. present three Pareto-based multi-objective GP methods that evolve decision tree like classifiers in [114]. The authors utilize a modified version of POPE-GP, where all individuals in the population are ensured to be unique with respect to classification accuracy and expression size. The authors introduce the 'Decomposed Multi-Objective GP' (DECMO-GP) technique which introduces the error rate on each

class as a separate objective. The goal of this modification is to evolve classifiers that successfully learn all groups present in the data which can also be a useful direction in the case of unbalanced datasets. However, the authors do not explicitly pursue this goal in their paper. Similar to the modified version of the POPE-GP, all individuals in the population are unique with respect to the values of the multiple objectives and the ties are broken in random. The third technique presented in the paper is called the 'Decomposed Multi-Objective-Parsimony GP' (DECMOP-GP) where the ties are broken in favor of individuals with smaller expression trees. The results are presented on three benchmark datasets that contain two, three and four groups. According to the authors, all three multi-objective schemes outperform the standard GP implementation with respect to classification accuracy and compactness of the solutions.

The multi-objective GP based feature extraction method presented by Zhang and Rockett in [163, 164] utilizes the SPEA2 algorithm with the tree size as an explicit objective. The multi-objective data visualization method presented by Valdes et al. in [151, 150] is based on the NSGA-II multi-objective optimization framework with two objectives related to the quality of the visualizations. Since their method does not utilize a variable-length tree representation, expression size is not an issue to be considered. Smith's feature extraction method [138] is based on the POPE-GP algorithm mentioned above. More details on these three feature extraction methods have been presented in section 5.1.

In genetic programming, the use of multi-objective schemes has been especially motivated as an alternative method to control bloat and evolve compact but highly 'successful' programs for various tasks. However, it has been argued that using the expression size as an explicit objective would lead to a phenomenon that is known as *premature convergence* or *population collapse* which manifests itself as the event of very simple but unfit individuals taking over the population at earlier stages of the evolutionary process [52, 76]. After that point, recovery is highly unlikely. This can be seen as the opposite of code bloat and it is equally, if not more, undesirable for GP.

DeJong and Pollack provide an in depth discussion of this phenomenon including the symptoms, possible reasons and solutions in [76]. The authors investigate the use of the tree size as a secondary

objective on the '6-parity' benchmark problem. Upon empirical analysis of the problem, they report that at each generation, the distribution of newly generated non-dominated trees is highly skewed towards sizes that are smaller than the average tree size in the population. Consequently, the selection algorithm is more likely to pick the trees with smaller sizes for breeding to prepare the next generation. As generations advance, the population collapses to trees with minimal size but low fitness. The authors argue that decreasing population diversity is the cause of this issue. They first investigate a diversity maintenance method which preserves population diversity by removing duplicate individuals with respect to fitness and tree size (unique phenotypes). The resulting algorithm produces significantly better results on three test problems including one instance of symbolic regression. Then, the authors propose another algorithm to actively promote population diversity by making diversity as an explicit measure. The algorithm is called 'Find Only and Complete Undominated Sets' (FOCUS), which is a highly elitist method that keeps only the non-dominated individuals in the population. Fitness, size and diversity are the three objectives considered where each individual's contribution to population diversity is measured in terms of an edit distance where highly unique individuals receive a higher diversity score (unique genotypes). Along with diversity maintenance through removal of redundant individuals, the authors report significantly better results with highly compact evolved trees.

Badran and Rockett present an alternative view in [14]. The authors argue that DeJong and Pollack's initial findings on how population collapse happens is largely due to the lack of mutation operator in their experiments. They present a set of experiments where inclusion of the mutation operator help preserve population diversity and prevent population collapse. Common multi-objective optimization schemes, such as the SPEA2 and NSGA-II methods are designed to preserve population diversity regardless of the nature of the multiple objectives. Moreover, Badran and Rockett claim that mutation operator is also beneficial within these frameworks when tree size is used as one of the objectives. However, high mutation rate can have adverse effects on the evolutionary process. Population collapse is also reported by Parrott et al. [114] when POPE-GP (which is based on NSGA-II) is used in order to evolve data classifiers where accuracy and tree size are the two objectives. Similar to the diversity maintenance

approach by DeJong and Pollack, the authors propose an elimination step where duplicate individuals with respect to accuracy and tree size are removed from the population before the breeding stage for the next generation begins.

Table 6.1 presents a summary of the alternative approaches discussed in this section. Each method possesses certain advantages and disadvantages.

Approach	**Pros.**	**Cons.**
(I) Weighted combination	(1) Simplicity	(1) Ad-hoc assignment of weights, (2) non-commensurable criteria, (3) incompatible units of measurement of the objectives, (4) One optimal solution is found, multiple runs are needed for more solutions
(II) Prioritizing the objectives	(1) No need to combine non-commensurable objectives into one scalar fitness value and search for optimal weights	(1) Objectives are needed to be ordered with respect to importance to the user, (2) a tolerance parameter is needed to adjust the trade-off between the objectives, (3) One optimal solution is found, multiple runs are needed for more solutions
(III) Prioritizing the objectives along with explicit complexity control	In addition to (II), separation of the complexity from other objectives reduces the number of objectives to prioritize	(1) In addition to (II), setting the parameter(s) for the complexity control method (such as the Tarpeian method for complexity control)
(IV) Pareto-Based approach	(1) No need to specify importance between the objectives, (2) no need to pre-determine trade-off between the objectives through weights or tolerance parameters, (3) a richer set of optimal solutions representing different trade-offs can be found within one run	(1) Multiple optimization frameworks such as: SPEA2, NSGA-II and others to consider, (2) since multiple optimal solutions are found, the burden is on the user to choose one final solution to use in decision making, (3) using size as an explicit objective requires precautions to maintain population diversity in order to avoid population collapse (premature convergence)

Table 6.1: Comparison of alternative methods for implementation of a GP based multi-objective optimization application

As it was also pointed out by Freitas [56], the weighted combination is an ad-hoc technique which requires the user to determine the trade-off between the objectives before the run. Likewise, the prioritization based approach requires the user to order the objectives with respect to perceived importance along with a tolerance value which is practically another way of formulating a trade-off. For G3P experiments presented in the last chapter, we had skipped the tolerance value in favor of simplicity. Moreover, if the user can not decide on an optimal trade-off, multiple experiments with different values are needed.

A Pareto-based approach does not force the user to make a priori choices about the relative importance of the objectives or the trade-offs. Instead, the algorithm explores solutions that represent different trade-offs within one run and lets the user make choices after the search is concluded. Facing

multiple solutions for a problem can be challenging as well as rewarding. Though it can be argued that multiple runs of a single objective implementation can be comparable to a Pareto-based approach, there is no guarantee that solutions generated during different runs will form the same (or a better) Pareto set.

6.2 The MOG3P Algorithm

In this section, we present a Pareto-based implementation of the G3P algorithm which we call 'Multi-Objective Genetic Programming Projection Pursuit' (MOG3P). Our implementation is based on the SPEA2 optimization framework. Algoritm 12 presents the workflow of MOG3P. The implementation is similar to G3P (section 5.3) with modifications on the fitness computation and individual selection components (highlighted in red).

Algorithm 12: The MOG3P algorithm for visual data classification

Input : NxR data matrix D
Set of functional symbols B =$\{b_1, b_2, ..., b_K\}$
Population size I, Archive size H,
Fitness Function $SPEA2\_F$
Classifiability Criterion C
Visualization Criterion V
Semantic Criterion S
Number of transformation expressions in T(2)

Output: A*: non-dominated set of solutions in final archive A_{final} consisting of NxR transformed data matrix M, data transformation expressions T

Randomly create initial population (P_0) of transformation functions (expressions) T
compute $SPEA2\_F(M|D, C, V, S)$ for population P_0
repeat
  BuildArchive A_g
  Select the transformations (T) from A_g as parents
  Perform breeding (crossover, reproduction, mutation operations) to create offsprings
  compute $SPEA2\_F(M|D, C, V, S)$ for population P_g
until *maximum number of generations are completed or an ideal solution is found*

The main difference is that the fitness of each individual is no longer independent from other members of the population. Fitness assignments are performed based on the Pareto-dominance relationships between the individuals. Our implementation is based on the ECJ package where the archive is maintained as a part of the population that contains the 'elite' individuals.

Algorithm 13 presents the computation of the fitness values for each individual in the population. For

each individual, both dominating and dominated solutions are taken into account. The strength of an individual is determined by how many individuals it dominates. Then the raw fitness of each individual is computed as the sum of the strengths of its dominators in the whole population. In the case when most individuals do not dominate each other, the raw fitness would not be a good measure to compare the individuals.

Density information is introduced as an additional measure to help discriminate between the individuals with identical raw fitness values. The density for each individual is computed in terms of the distance between the individual and its k-th Nearest Neighbor. The distance measure is the Euclidean distance between the objective vectors representing each individual and smaller distance means higher density. For the ECJ implementation k is set to $\sqrt{|P_g|}$. Finally, the fitness of an individual is computed as the sum of its raw fitness and density information. Here, the fitness is minimized, therefore the individuals with smaller sum of raw fitness and density information are considered better than others. The raw fitness contributes to the quality of the solutions while density information aims to promote diversity.

Algorithm 13: MOG3P SPEA2F

Input : Current population P_g,
NxR data matrix D
Classifiability Criterion C
Visualization Criterion V
Semantic Criterion S

Output: Fitness values SPEA2F for all individuals in P_g

//Compute the strength of each individual as the number of individuals it dominates **foreach** *transformation T in population* **do**
  M=transform(D,T) ;
  v_T = ($C_T(M)$, $V_T(M)$, S_T) ;
  $Strength(T)$ = $|\{t|t \in P_g \wedge dominates(v_T, v_t)\}|$;
end
//Assign fitness to each individual
foreach *transformation T in population* **do**
  $RawFitness(T)$ = $\sum_{T' \in P_g, dominates(v_{T'}, v_T)} Strength(T')$;
  //Density is computed based on the k-Nearest Neighbors algorithm
  $DensityFitness(T)$ = $\frac{1}{\sigma_T^k + 2}$;
  $SPEA2F(T)$ = $RawFitness(T)$ + $DensityFitness(T)$;
end

Algorithm 14 summarizes the process of building the archive which is the group of 'elite' individuals that drive the evolutionary process. At each iteration, the non-dominated individuals in the current population is moved to the archive. There are two possible cases. If the archive is full beyond its limit it is truncated. The truncation works in a way that the Pareto set is decimated with respect to the

density. At each iteration, one individual with minimum distance to others is removed until the archive is truncated to the predefined size. In the second case, if the archive is not full, a number of best dominated individuals according to their computed fitness values are moved to the archive until it is filled.

Algorithm 14: MOG3P BuildArchive

Input : Population P_g, Archive Size H
Output: Archive A_g

//Method: No, Unique Genotypes, Unique Phenotypes (objective vectors)
P_g=EliminateDuplicates(P_g,method);
//Insert all non-dominated individuals of P_g into A_g
A_g= ParetoFront(P_g) ;
if $|A_g| > H$ **then**
  A_g=Truncate(A_g);
else
  //Include dominated individuals until $|A_g|$=H
  A_g=FillArchive(A_g, P_g-ParetoFront(P_g),H) ;
end

Although the initial GP tree generation algorithm prevents duplicate individuals, the general SPEA2 implementation does not enforce uniqueness. As it was mentioned earlier in this chapter, elimination of identical individuals has been found to be helpful in preventing the population collapse such as in POPE-GP for evolving decision tree like classifiers. We incorporated an optional duplicate elimination step in the archive building process where the duplicate individuals are eliminated with respect to the genotype (the expression trees) or the phenotype (the values of the three objectives).

6.3 MOG3P Experiments

In this section, we first devise and compare different experimental configurations for the MOG3P and then present comparisons between the MOG3P and G3P results from last chapter. We study the MOG3P runs from the perspectives of population collapse, diversity and fitness and investigate how these concepts relate to each other for our problem domain. The single objective and multi objective implementations are compared with respect to the quality of the solutions found using each scheme.

6.3.1 Population Collapse, Diversity and Fitness

As we mentioned earlier, population collapse has been a common complaint when expression size was used as one of the objectives in a multi-objective optimization scheme. Genotype and Phenotype based duplicate elimination methods have been widely proposed as remedy. In this section, we study the effects of the duplicate elimination schemes on the MOG3P from the perspective of population collapse, diversity and fitness. We also compare the two objective (classifier accuracy, expression size) implementation of MOG3P to the three objective (classifier accuracy, visual quality measure, expression size) implementation.

The following table shows the settings used in these experiments. Unless otherwise stated all experiments presented in this chapter were conducted using this configuration.

GP Type	Generational
Population Size	200
Archive Size	100
Generations	50
Crossover Operator	Subtree Crossover
Crossover Probability	0.9
Reproduction Probability	0.05
Mutation Operator	Swap Mutate
Mutation Probability	0.05
Fitness Assignment	SPEA2
Selection Algorithm	SPEA2 Tournament Selection
Tournament size	7
Functional Symbols	$\{+, -, *, protected/, min, max, power, log\}$
Ephemeral Random Constant (ERC)	$[-1, 1]$
Initialization	Ramped half-and-half (minimum depth:2, maximum depth:6)
Maximum tree depth	17
Classifiability Objective (C)	Random (Naive Bayes, C4.5 (J48), K-Nearest Neighbors)
Visualization Objective (V)	T=2, Median Vis.
Semantic Objective (S)	I_{TS}
Evaluation	10 times 10-fold cross validation (total 100 runs)

Table 6.2: MOG3P Settings

The percentage of the population with minimal sized expression trees (single node) has been used as an indicator of population collapse in [14]. We set up six experiments with different settings for the objectives and the population duplicate elimination methods (shown below).

Figure 6.1 shows the effects of these different settings on the population collapse for three datasets. Each plot shows the average of the percentage of the simplest trees (one node tree per data trans-

Experiment	Objectives	Duplicate Elimination
All Baseline	Random Classifier Policy, Median Vis., I_{TS}	None
All Phenotype	Random Classifier Policy, Median Vis., I_{TS}	Phenotype
All Genotype	Random Classifier Policy, Median Vis., I_{TS}	Genotype
No Vis. Baseline	Random Classifier Policy, I_{TS}	None
No Vis. Phenotype	Random Classifier Policy, I_{TS}	Phenotype
No Vis. Genotype	Random Classifier Policy, I_{TS}	Genotype

Table 6.3: MOG3P Population Collapse Experiments

formation expression) at each generation across all 100 runs. The re-occurring pattern between the

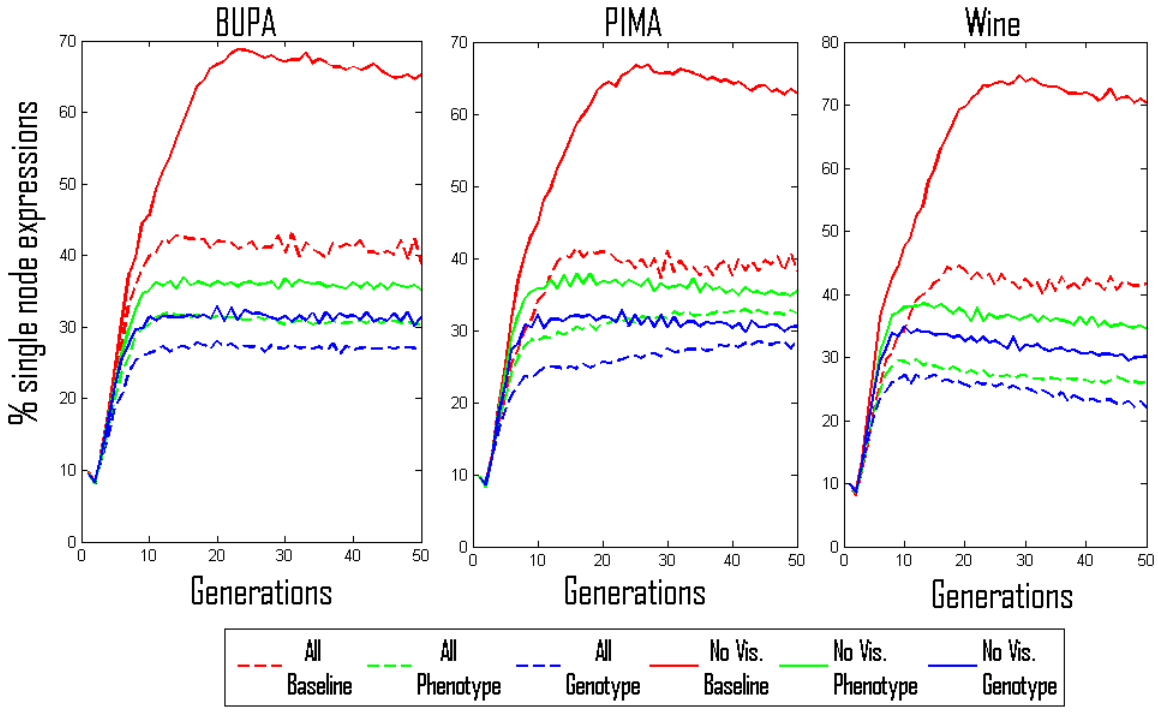


Figure 6.1: Population Collapse on the BUPA, PIMA and Wine datasets

runs is that the population collapse is much higher when no duplicate elimination method is utilized as opposed to incorporating either the genotype or the phenotype based elimination. Note that, phenotype based elimination covers the genotype based elimination since the individuals with identical expressions trees are guaranteed to have the same phenotypes. It is also observed that even without duplicate elimination, using all three objectives together as opposed to only two objectives (ignoring the visualization objective) helps prevent population collapse. The 'No Vis. Baseline' option is by far the worst in terms of population collapse which reaches as high as around %70 during the evolutionary process. All other options show considerably less population collapse while it is evident that duplicate

elimination help reduce population collapse especially with the inclusion of the visualization objective.

As it was mentioned above, it has been widely argued that population collapse was related to the loss of diversity in the population. Here, we investigate relationships between population diversity and population collapse on the MOG3P experiments presented in figure 6.1. We consider two types of diversity: genotype based (genotypic entropy), where the structural diversity of the expression trees are considered and phenotype based (phenotypic entropy), where the functional diversity (behavior based fitness) of the expression trees are considered. We compute these diversity measures in terms of entropy that was introduced into the GP literature by Rosca in [126] and further explored by Tomassini et al. and Burke et al. in [146] and [31] respectively. Entropy is computed as follows:

$$-\sum_{k=1}^{K} = p_k \; ln(p_k)$$

where p_k is a partition of the population sharing the same characteristic.

In genotypic entropy, p_k is defined as the ratio of the population sharing the same genotype. In MOG3P, each individual contains two expression trees, corresponding to the data transformation functions for visualization. Before calculating genotypic entropy, we sort the expression trees of each individual in lexicographic order so that the order of the expressions would not matter when comparing two individuals (for example [weight * height, depth] becomes [depth, weight * height] after sorting).

In phenotypic entropy, p_k is defined as the ratio of the population sharing the same objective fitness vector. In the first three experimental settings shown in table 6.3, the objective fitness vectors contain all objectives (classifiability and visual interpretability and feature complexity). For the remaining three experiments, the visual interpretability measure is not included in the objective fitness vectors.

High entropy indicates that most individuals in the population are different from each other where low entropy means that a lot of individuals share the same characteristics. Tracking the changes in entropy helps us understand the population dynamics during the evolutionary process.

Figure 6.2 presents the average values of genotype and phenotype diversity at each generation

across all 100 runs for all six experimental settings. Similar trends are observed across the results on these three datasets. Rapid and large decrease in genotypic entropy is observed for those experiments without the duplicate elimination.

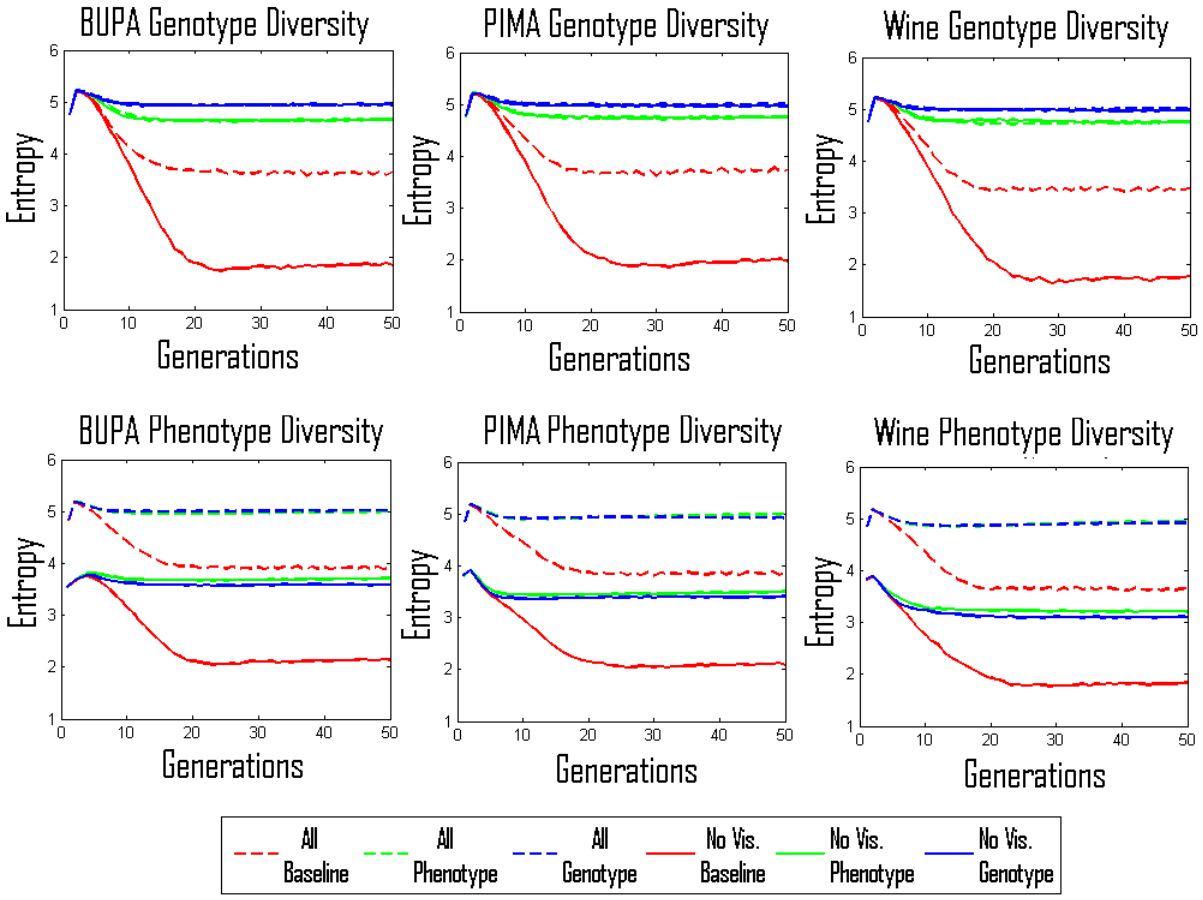


Figure 6.2: Entropy based measures of genotype and phenotype diversity

Elimination of duplicates based on either the genotype or phenotype seems to preserve population diversity. As far as the phenotypic entropy is concerned, the same observation holds; the experiments with duplicate elimination show that diversity is preserved during the course of the evolutionary process while the baseline results indicate rapid and large loss of phenotypic diversity. There is one main difference between the genotypic and phenotypic diversity plots. Genotypic entropy of the initial populations in all six experimental settings are very similar while the phenotypic entropy of the initial populations are significantly lower when the visual interpretability objective is omitted from the phenotype. Without

duplicate elimination, the overall drop in phenotypic diversity is much smaller when all three objectives are used.

The findings on the relationship between diversity and population collapse can be summarized as follows:

- Based on the plots, the larger the loss in diversity is, the higher the population collapse gets.
- All other experimental conditions being the same, explicit measures for diversity preservation (elimination of duplicates) demonstrate smaller loss in diversity and population collapse compared to utilizing no such measures. We find no significant difference between eliminating the duplicates based on the genotype or the phenotype.
- When classifier accuracy and expression size are used as the two objectives as opposed to the three objectives, the genotypic entropy of the initial populations are close. However, phenotypic entropy in the case of two objectives is considerably lower. Classifier accuracy and size are discrete measures where many different genotypes might correspond to same values. Visual interpretability is more sensitive to small differences in the generated visualization thus increasing phenotype entropy. Incorporating all three objectives without duplicate elimination demonstrates comparable results to all other settings except that it is much superior to using only two objectives.

We now focus on the MOG3P from the perspective of fitness and investigate how the observed diversity and population collapse might be related to fitness. Figures B.2- 6.5 show MOG3P performance on the BUPA, PIMA and Wine datasets based on classification accuracy on the training set, test set [1], visual interpretability and feature complexity (total expression size). For the accuracy and median vis measures, the best values and for the tree size, the average value is plotted at each generation.

Based on the MOG3P results presented here, it can be seen that the two-objective option without duplicate elimination ('No Vis. Baseline') offers the worst average performance (based on training set

[1] As in the last chapter, the test sets are not used in any way to drive the evolutionary process. At each generation, the individuals with best training set accuracy are evaluated on the test set and the best test set accuracy is recorded only for the purpose of generating the plot.

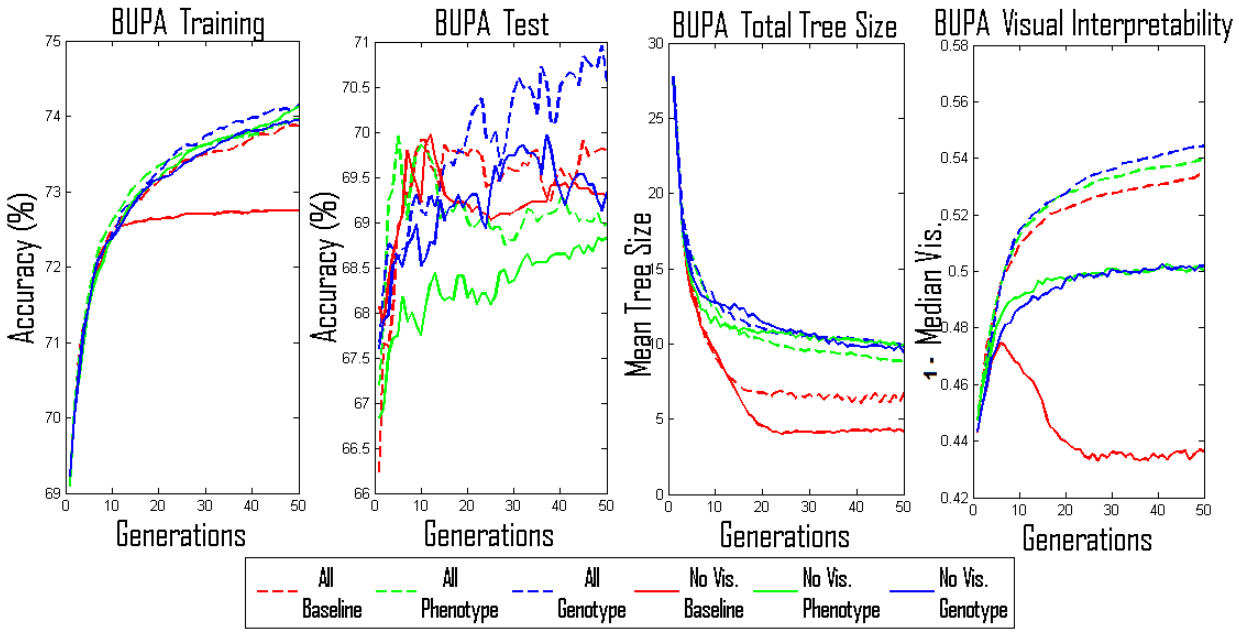


Figure 6.3: MOG3P results on BUPA dataset

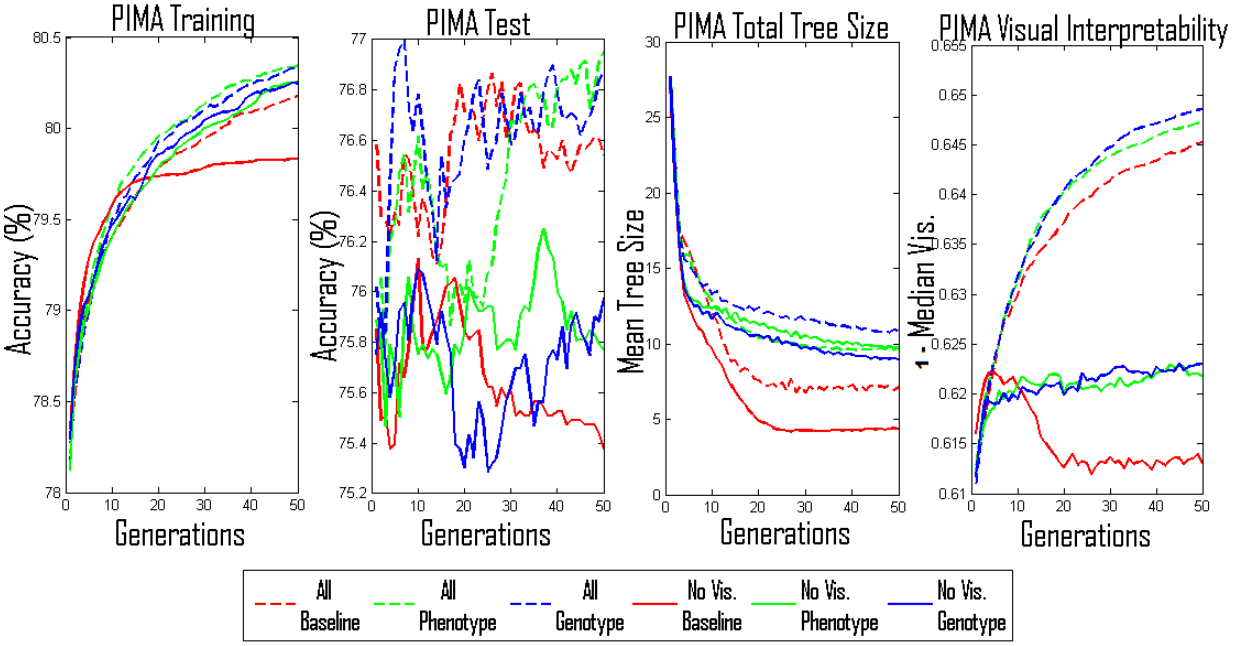


Figure 6.4: MOG3P results on PIMA dataset

accuracy and visual interpretability) which is parallel to the very high population collapse and loss of diversity. However, we refrain from claiming that the difference in performance was actually *caused* by population collapse or loss of diversity.

The reported cases of population collapse or premature convergence to very small expression trees

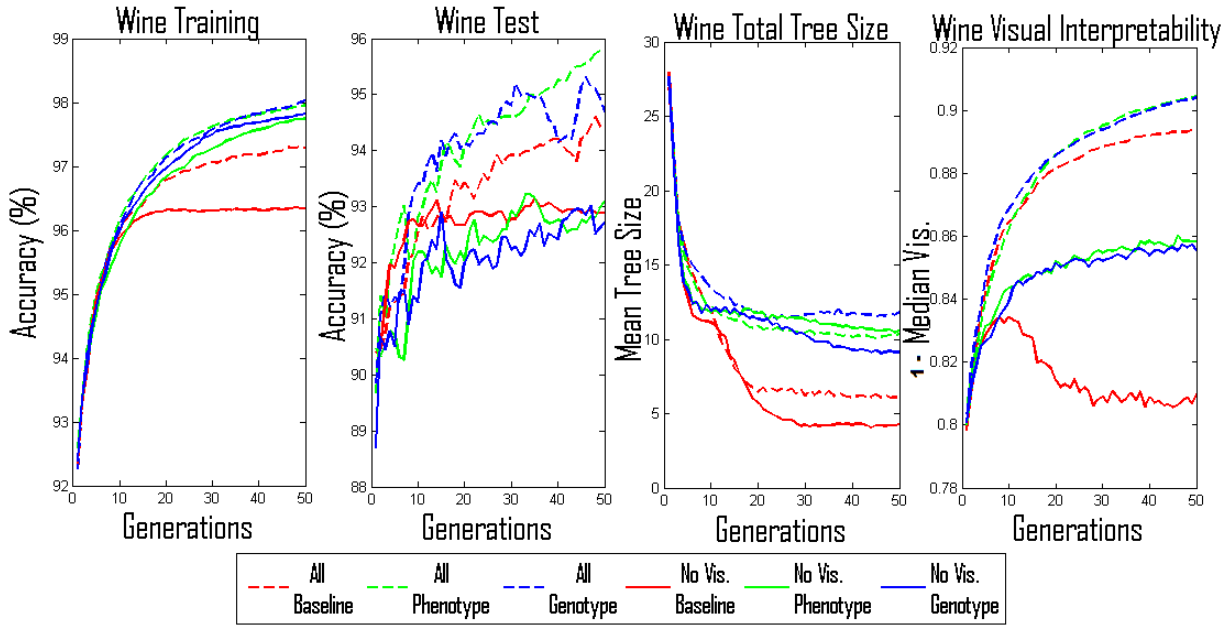


Figure 6.5: MOG3P results on Wine dataset

are generally within the context of evolving GP based classifiers (such as those mentioned in section 6). As it is also emphasized by Smith in [138], there is a difference between using GP to generate decision tree like classifiers and to generate new features. While very simple decision trees might fail to classify the data, it is possible to achieve high classification rates even with very simple features using any classification algorithm. The MOG3P results also confirm this intuition. We observe that even with high population collapse and very simple expression trees, the classification results are not drastically low.

The visual interpretability plots reveal that two objective optimization without the inclusion of the visual interpretability objective prevents the MOG3P from generating high quality visualizations as measured by the median of all six visual quality measures presented in the last chapter. This observation is of course not surprising, however, it shows that even though classifier accuracy is related to the quality of the visualizations, they do not necessarily optimize compactness versus separation as measured by a number of explicit measures.

Tables 6.4- 6.6 represent the best-of-run MOG3P results. For each dataset, the mean and standard deviations of the four measures across all 100 runs are reported using each of the six experimental configurations. Higher accuracy indicates better classifiability, lower median vis. indicates views with

better quality and lower total tree size indicates simpler feature transformation expressions.

We follow the following procedure in testing and reporting statistical significance in comparing these results: on each dataset, the effect of duplicate elimination (none, phenotype-based, genotype-based) is tested on those experiments using the same set of objectives (All objectives or all but visual interpretability) and the effect of using two or three objectives is tested on those experiments using the same duplicate elimination method. Specifically, the effect of omitting the visualization objective is tested on pairs 'All Baseline'-'No. Vis. Baseline','All Phenotype'-'No. Vis. Phenotype','All Genotype'-'No. Vis. Genotype' separately using the corrected t-test at $\alpha = 0.05$. The effect of the duplicate elimination method is studied as a series of 3 pairwise tests on 'All' and 'No Vis' experiment results at $\alpha = 0.05/3$.

Method	Training Set Accuracy (%)	Test set Accuracy (%)	Median Vis.	Mean Total Tree Size
All Baseline	73.87 (1.19)	69.79 (6.64)	($\surd$) 0.47 (0.020)	6.73 (3.42)
All Phenotype	73.96 (1.25)	68.98 (7.07)	($\surd$) 0.46 (0.019)	8.84 (3.16)
All Genotype	74.15 (1.08)	70.54 (6.56)	($\surd$) 0.46 (0.017)	9.70 (3.01)
No Vis. Baseline	72.75 (1.37)	69.29 (6.29)	0.56 (0.043) (x)	($\surd$) 4.19 (1.52) ($\surd$)
No Vis. Phenotype	74.12 (1.40)	68.82 (6.84)	0.50 (0.037)	9.95 (5.29)
No Vis. Genotype	73.94 (1.39)	69.33 (6.15)	0.50 (0.034)	9.44 (3.53)

Table 6.4: Comparison of the MOG3P results based on the final population on BUPA dataset

Method	Training Set Accuracy (%)	Test set Accuracy (%)	Median Vis.	Mean Total Tree Size
All Baseline	80.18 (0.73)	76.54 (5.52)	($\surd$) 0.35 (0.009)	7.09 (1.88)
All Phenotype	80.34 (0.73)	76.95 (5.54)	($\surd$) 0.35 (0.008)	9.79 (2.98)
All Genotype	80.33 (0.71)	76.88 (5.41)	($\surd$) 0.35 (0.008)	10.80 (3.54)
No Vis. Baseline	79.83 (2.16)	75.38 (5.50)	0.39 (0.041)	($\surd$) 4.35 (1.83) ($\surd$)
No Vis. Phenotype	80.26 (0.74)	75.78 (5.62)	0.38 (0.011)	9.63 (3.43)
No Vis. Genotype	80.25 (0.79)	75.98 (5.18)	0.38 (0.013)	8.95 (3.78)

Table 6.5: Comparison of the MOG3P results based on the final population on PIMA dataset

Method	Training Set Accuracy (%)	Test set Accuracy (%)	Median Vis.	Mean Total Tree Size
All Baseline	97.29 (1.05)	94.49 (5.23)	($\surd$) 0.11 (0.017)	6.09 (2.00) ($\surd$)
All Phenotype	97.95 (0.85)	95.83 (4.63)	($\surd$) 0.10 (0.016)	10.39 (3.77)
All Genotype	98.04 (0.80)	94.70 (5.92)	($\surd$) 0.10 (0.018)	11.89 (5.00
No Vis. Baseline	96.34 (1.45)	92.89 (6.23)	0.19 (0.05) (x)	4.29 (2.12) ($\surd$)
No Vis. Phenotype	97.75 (1.03)	93.09 (5.90)	0.14 (0.03)	10.56 (4.38)
No Vis. Genotype	97.83 (1.04)	92.73 (6.46)	0.14 (0.02)	9.14 (4.04)

Table 6.6: Comparison of the MOG3P results based on the final population on Wine dataset

The $\surd$ signs on the left-hand side of a specific measure indicates statistically significant superiority of including the visual interpretability objective. The $\surd$ or (x) signs on the right-hand side of a specific measure indicates statistically significant superiority/inferiority of the duplicate elimination method em-

ployed within the two or three objective experiments. According to the results on these three datasets, we find that there is no significant difference in terms of the classification accuracy. Including visual interpretability as an explicit measure significantly increases the visualization quality. The duplicate elimination method has direct effects on the total tree size. Lack of duplicate elimination tends to generate smaller trees on the average. However, the difference is only significant in the two-objective experiments. When all three objectives are used, we find that significantly smaller trees are generated only on Wine dataset.

In summary, based on the empirical results presented here, we find that population collapse does not pose a threat within the context of using GP for feature extraction as opposed to the reported cases where GP is used to evolve decision tree like classifiers. We did not find significant increase in classifier accuracy with the genotype and phenotype based duplicate elimination techniques on our problem domain. On the contrary, the duplicate elimination step imposes additional computational burden on two levels: the cost of duplicate elimination itself and the cost of evaluating larger trees. Therefore, all experiments after this point are conducted using the 'All Baseline' configuration.

6.3.2 Comparison to G3P

In this section, we compare the MOG3P to two G3P implementations we presented in the last chapter. First we compare a number of performance measures based on the final population (best-of-run) individuals. All values presented in the following table are average measures across all 100 runs using each method. For the G3P-linear combination, all three objectives are combined into a scalar fitness value using weights $\alpha = 0.5$,$\beta = 0.5$ and $\gamma = 0.1$ for the classifiability, median of visual interpretability and total tree size as the feature complexity. The G3-Lexicographic selection $+$ Tarpeian Method uses the same weights for the classifiability and visualization objectives and feature complexity is controlled using the Tarpeian method which penalizes over-average complexity individuals. Each G3P run returns one best-of-run solution where the MOG3P returns multiple best-of-run solutions representing a different trade-off between the objectives. In order to compare the MOG3P to G3P, for each MOG3P run, we

choose the individual with the best training set accuracy and report the corresponding test set accuracy. In case of ties, the best set set accuracy is selected. For the visual interpretability, we report the best median vis. measure encountered for each MOG3P run. Finally, the total nodes evaluated measure gives an idea about the computational burden of evaluating expression trees accumulated through the course of the run from the initial generation till the end.

Dataset (Method)	Training Set Accuracy (%)	Test set Accuracy (%)	Median Vis.	Total Nodes Evaluated
BUPA (G3P)- Linear Combination	72.52 (1.79)	68.33 (7.26)	0.49 (0.030)	55,150.5 (21,431.3)
BUPA (G3P)- Lexicographic selection + Tarpeian Method	74.45 (1.84)	69.95 (7.10)	0.47 (0.026)	177,574 (81,350.2)
BUPA (MOG3P)	73.87 (1.19)	69.79 (6.64)	0.47 (0.020)	42,748 (10,269.3)
PIMA (G3P)- Linear Combination	79.07 (0.88)	75.36 (5.30)	0.38 (0.020)	32,839.3 (9,445.44)
PIMA (G3P)- Lexicographic selection + Tarpeian Method	80.61 (0.99)	76.07 (5.26)	0.36 (0.010)	157,788 (83,729.5)
PIMA (MOG3P)	80.18 (0.72)	76.54 (5.52)	0.35 (0.010)	47,449.3 (10,767.7)
Wine (G3P)- Linear Combination	96.09 (1.40)	91.82 (6.42)	0.12 (0.030)	53,899.8 (19,046.3)
Wine (G3P)- Lexicographic selection + Tarpeian Method	97.29 (1.43)	93.99 (5.88)	0.10 (0.028)	182,130 (101,443)
Wine (MOG3P)	97.29 (1.05)	94.49 (5.23)	0.11 (0.017)	44,398.2 (10,446.9)

Table 6.7: Comparison of MOG3P best-of-run results to two implementations of G3P

In terms of the training set accuracy, test set accuracy and visualization objectives we did not find significant differences between these three implementations. However, the G3P with Tarpeian method spends significantly more computational effort indicating it could have been more aggressive in through different parameter values (see section 5.9.4.1). The computational effort for the G3P-linear combination runs are comparable to the MOG3P. Therefore, we will consider this implementation to further compare with the MOG3P.

Figures 6.6- 6.8 show the best-of-run individuals of all 100 runs using the MOG3P and the G3P-linear combination methods. For each algorithm, the optimization is based on three objectives: classifiability on training set, visual interpretability and feature complexity. After the best of run individuals are identified, their test set performances are computed and reported.

In these plots, we present the results with respect to two trade-offs (as in section 5.9.5). The classification performance trade-off is based on the training and test set accuracy and the interpretability trade-off is based on the median vis. and feature complexity measures. We choose this way of presenting the results, since it is a logical break-down of the objectives and much easier to visualize. [2] In each

[2] An alternative 3D visualization can be constructed where the axes represent training set accuracy, test set accuracy and

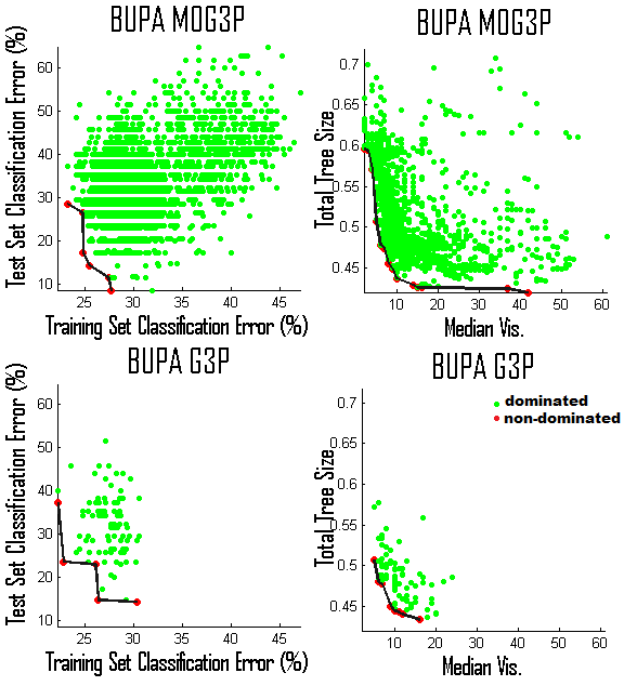


Figure 6.6: G3P-linear combination and MOG3P Pareto-sets on the BUPA dataset

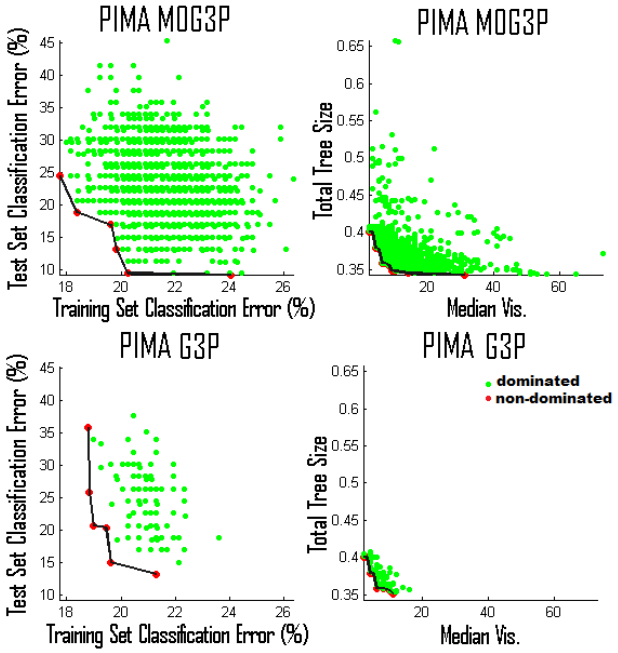


Figure 6.7: G3P-linear combination and MOG3P Pareto-sets on the PIMA dataset

figure, all best-of-run individuals are plotted and the Pareto-sets are computed and highlighted on the plots. The G3P plots contain only 100 points since each G3P run returns only one best-of-run solution.

median vis. measures and the points are sized according to the feature complexity. However, it would be much difficult to draw here and difficult for the users to inspect on the screen.

In MOG3P experiments, the number of best-of-run individuals varies since the trade-off between the three objectives are not predefined as in the G3P and the algorithm returns much more solutions based on the concept of Pareto dominance.

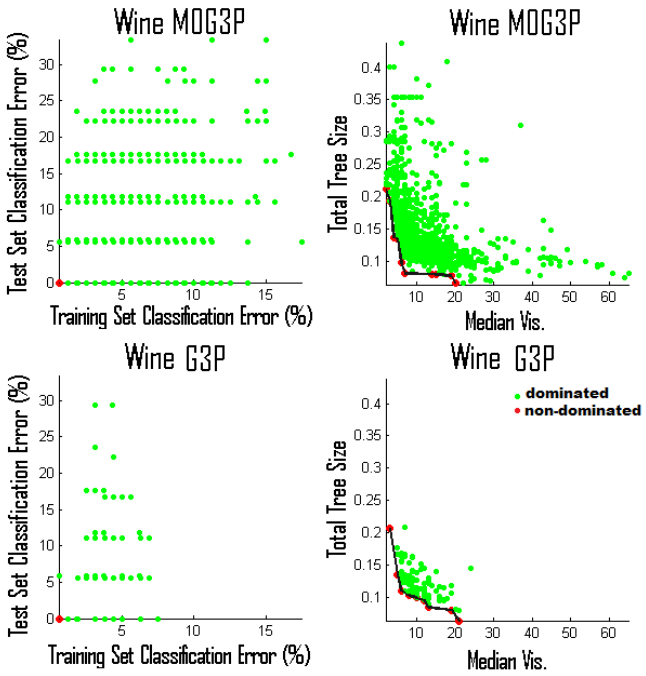


Figure 6.8: G3P-linear combination and MOG3P Pareto-sets on the Wine dataset

Each method provides a different Pareto set for the two trade-offs shown in the plots. Generally, the efficacy of a search algorithm is tested based on the closeness of the Pareto set to the true Pareto optimal set for the problem domain. However, for these data mining problems, no such ground truth exists. We compare the Pareto sets computed by the single objective (G3P) and multi-objective (MOG3P) implementations (figure 6.9). For each method, the number of dominating solutions are given on table 6.8. In almost all cases, the MOG3P finds better solutions compared to G3P.

Dataset (Method)	#Dominating Solutions Classifiability Trade-off	#Dominating Solutions Interpretability Trade-off
BUPA (G3P)- Linear Combination	2	0
BUPA (MOG3P)	4	13
PIMA (G3P)- Linear Combination	1	0
PIMA (MOG3P)	5	5
Wine (G3P)- Linear Combination	1	0
Wine (MOG3P)	0	9

Table 6.8: Comparisons of G3P and MOG3P generated Pareto sets in terms of Pareto dominance

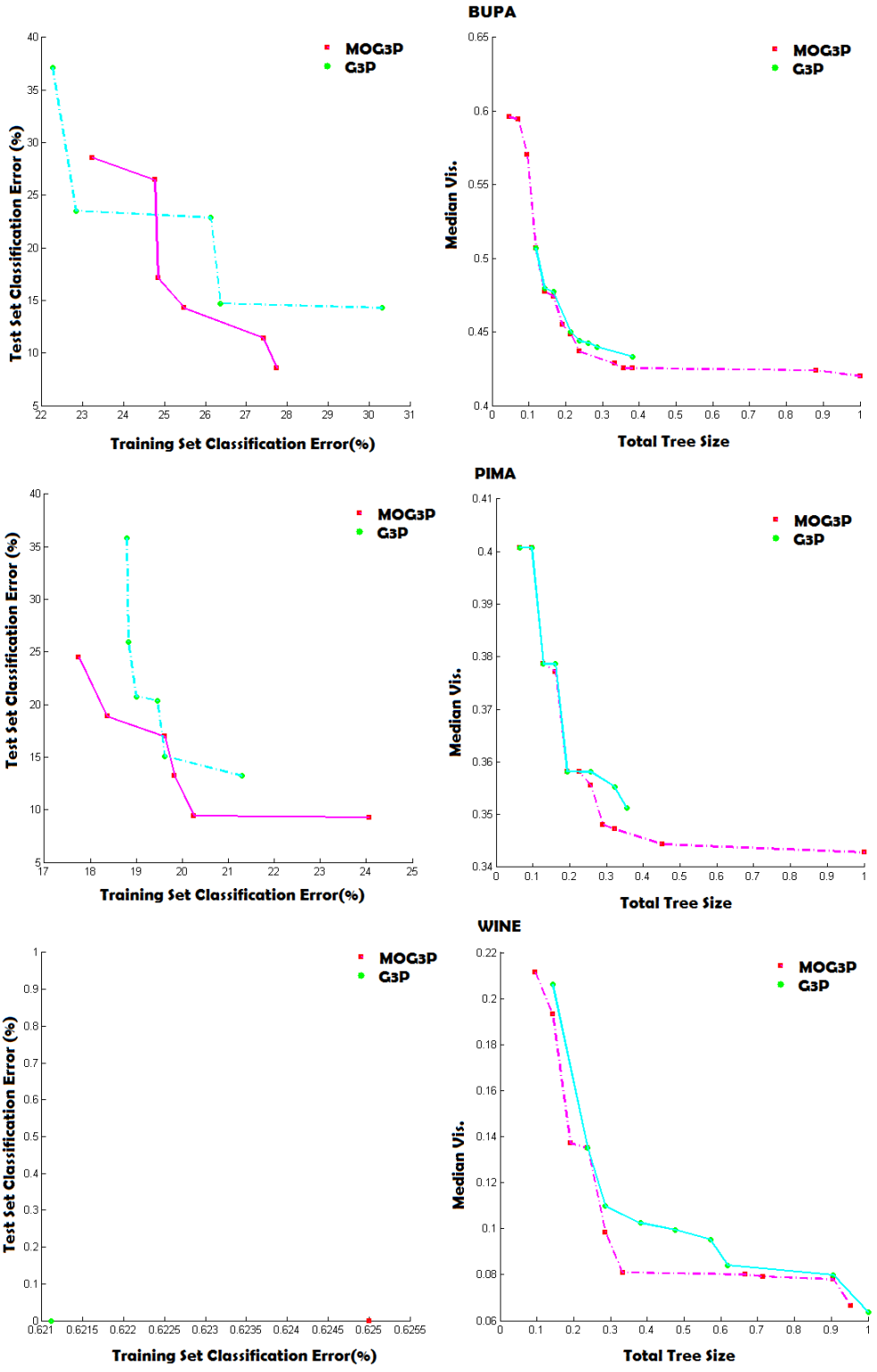


Figure 6.9: G3P versus MOG3P generated best solutions

6.3.3 Summary

In this section, we first examined six different experimental configurations for MOG3P where we studied the effects of using visual interpretability as a separate measure in addition to classification performance and feature complexity for feature extraction in visual classification and the effect of explicit measures to preserve genotypic and phenotypic diversity by duplicate elimination. Our experiments confirmed that incorporating visual quality measures as a third objective increases phenotypic diversity and without the inclusion of visualization measures, the quality of the views suffer in terms of separation and compactness of the groups.

Phenotype and Genotype based duplicate elimination techniques have been proposed in GP literature against the population collapse problem that occurs when the tree size is used as an explicit objective. We did not find a significant loss in the quality of the solutions in terms of accuracy and visual quality when no such duplicate elimination method was utilized. We observed increased average tree size when duplicate elimination was performed which indicates that duplicate elimination tries to pull the search towards more complex expression trees, however with no significant benefits in terms of the other objectives. We conclude that although population collapse has been an issue when GP is used to generate decision tree like classifiers, it is not that catastrophic in GP based feature extraction problem since even a pair of single attributes can be enough for successful classification using any classifier.

We then presented a comparison between the MOG3P and G3P results from the last chapter. The linear combination based G3P is found to be aggressively favoring parsimony (simplicity) where the Lexicographic selection with Tarpeian method was not aggressive enough in eliminating unnecessarily large individuals. Both implementations can be further optimized with extensive experimentation using different parameter settings. However, the MOG3P does not require any parameter settings to define trade-offs between the objectives, therefore saves us from the burden of determining the parameters that work well. Moreover, the Pareto sets of the solutions generated by MOG3P were found to be better than those generated by G3P in terms of closeness to the origin where all objectives are to be minimized. Overall, we conclude that pursuing a multi-objective implementation is more fruitful than a

single objective implementation in terms of the experimental process and the quality of the results.

6.4 Seeding in MOG3P

Most evolutionary data mining applications assume no domain knowledge related to the data analysis problem which can be considered as one of its strengths. However, domain knowledge or any known solutions can easily be incorporated into the algorithm in a number of ways. For instance, the expression trees can be constrained to a certain form such as 'sums of nonlinear functions of only three variables' or 'linear combinations of at most five variables'. This can be done through expression grammars [103] or using linear representations such as the gene expression programming [151], [150]. Another method which is called *seeding* incorporates domain knowledge or best known solutions into the evolutionary process by simply 'injecting' them into the population. Langdon and Nordin apply seeding in a number of GP based data classification problems [88]. A number of best known solutions on the training dataset are injected into the initial population. The authors report that the evolutionary process improves these initial solutions in terms of test set error and size of the expressions. Schmidt and Lipson utilize seeding in symbolic regression problems in [134] and [133]. Seeding in GP has also been applied to software engineering in [12] where the initial population was not random but contained the initial code where the goal is to evolve code that is functionally equivalent but more efficient in terms of execution time.

Seeding can be applied to MOG3P by incorporating best known data transformation expressions into the initial population. A good set of candidates are the 2-tuples of the initial data attributes. Amongst all possible 2D scatterplots that can be generated by pairing all original attributes, a number of 'good' views are selected using the same classifiability and visual interpretability measures as in MOG3P. These views indicate informative attributes where they can be used as a starting point to evolve more informative features. Another option is to incorporate the best solutions from the standard dimensionality reduction techniques into the MOG3P. As discussed in the last chapter, all three methods (PCA, MDA and TPP) generate weighted linear combinations of the original attributes as the visualization

axes for the generated views. By taking advantage of these results, the MOG3P acts as a hybrid of the standard and the evolutionary techniques for dimensionality reduction. The results of the standard methods can be further improved in terms of discriminative power and/or interpretability. For example, the evolutionary process can help reduce the expression size while maintaining its discriminative power.

6.4.1 Experiments

We present experiment results to compare the effect of seeding the initial population with the best known visualization of the data. In section 5.9.1, we have presented comparisons of G3P to three dimensionality reduction algorithms: PCA (Principal Components Analysis), MDA (Multiple Discriminants Analysis) and TPP (Targeted Projection Pursuit). On all datasets, except for the Wine dataset, the G3P was better or comparable to the standard methods with respect to the classifiability of the extracted pair of features with much smaller transformation expressions. On the Wine dataset, the visualization generated by the MDA method shows perfect separation (figure 6.10) using the following pair of transformation functions:

$$
\begin{aligned}
d_1 =& 0.4347 * ALC + -0.2441 * MACD + 0.2648 * ASH + -0.1627 * AASH + \\
& 0.0052 * MG + -0.4779 * TPEH + 1.5128 * FLV + \\
& 1.8888 * NFLV + -0.2291 * PROA + -0.3639 * CINT + \\
& 0.8673 * HUE + 1.2681 * DIL + 0.0021 * PROL \\
d_2 =& -0.7251 * ALC + -0.3402 * MACD + -2.2311 * ASH + 0.1396 * AASH + \\
& 0.0040 * MG + 0.0717 * TPEH + 0.3984 * FLV + \\
& 1.4508 * NFLV + 0.2967 * PROA + -0.2198 * CINT + \\
& 1.7069 * HUE + 0.117 * DIL + -0.0035 * PROL
\end{aligned}
$$

The total size of the MDA generated expressions is 102. Our hypothesis is that through seeding into MOG3P, we can reduce the size of these expressions while preserving the quality of the view. Seeding is performed by adding these two transformation functions into the initial population of the MOG3P runs.

Figure 6.11 shows three MOG3P generated visualizations. In plots (b) and (c), two best solutions generated using the seeding method are presented, in plot (a) the best solution without seeding is shown. The best solutions to present here were chosen in the following way: we first identified those solutions with the highest classifiability (highest training set and test set accuracy). In MOG3P without

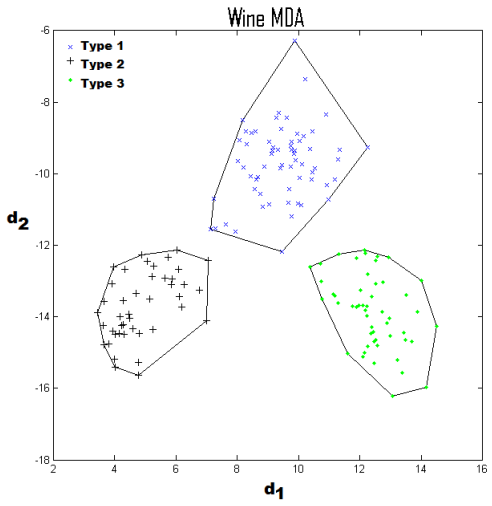


Figure 6.10: Best MDA generated 2D visualization of the Wine dataset

seeding, none of the solutions were 100% accurate but some of them were very close (%99.38). From the set of these best solutions, we then picked the smallest expressions with highest visualization quality.

As it can be seen from the plots, MOG3P using seeding finds visualizations with perfect separation while the visualization found without seeding shows (very litle) overlap between two groups.

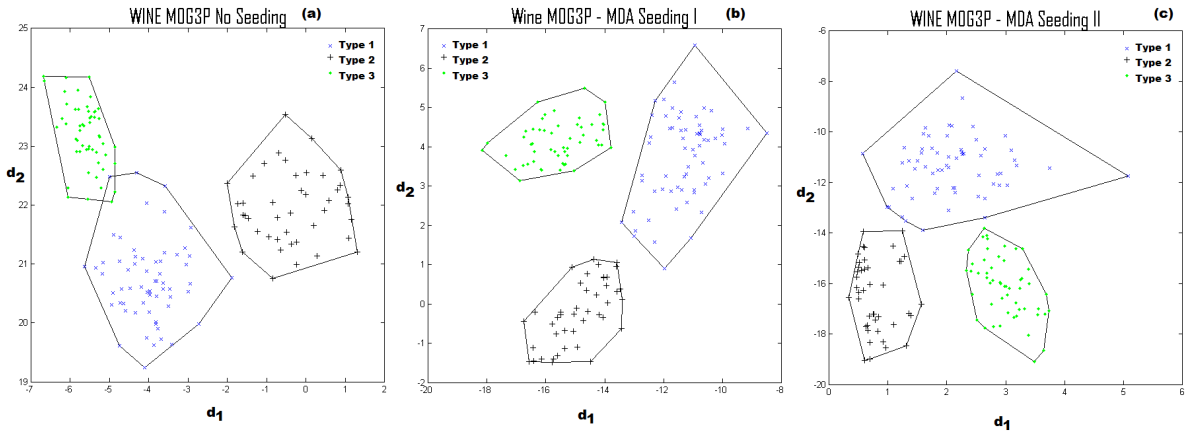


Figure 6.11: Wine data scatterplots generated using MOG3P with and without MDA seeding

The pair of expressions generating the visualization in plot (a) is given below:

$$
\begin{aligned}
d_1 =& a - (b - c) \\
a =& log(ASH - FLV) \\
b =& Min(ASH, log(FLV)) + log(PROL) \\
c =& \frac{0.701 - (FLV - \frac{CINT}{DIL})}{DIL} \\
d_2 =& ALC + log(PROL) + ASH
\end{aligned}
$$

Total size of these expressions is 28 and they contain only 6 of the 13 original attributes, reaching a %54 percent attribute elimination rate.

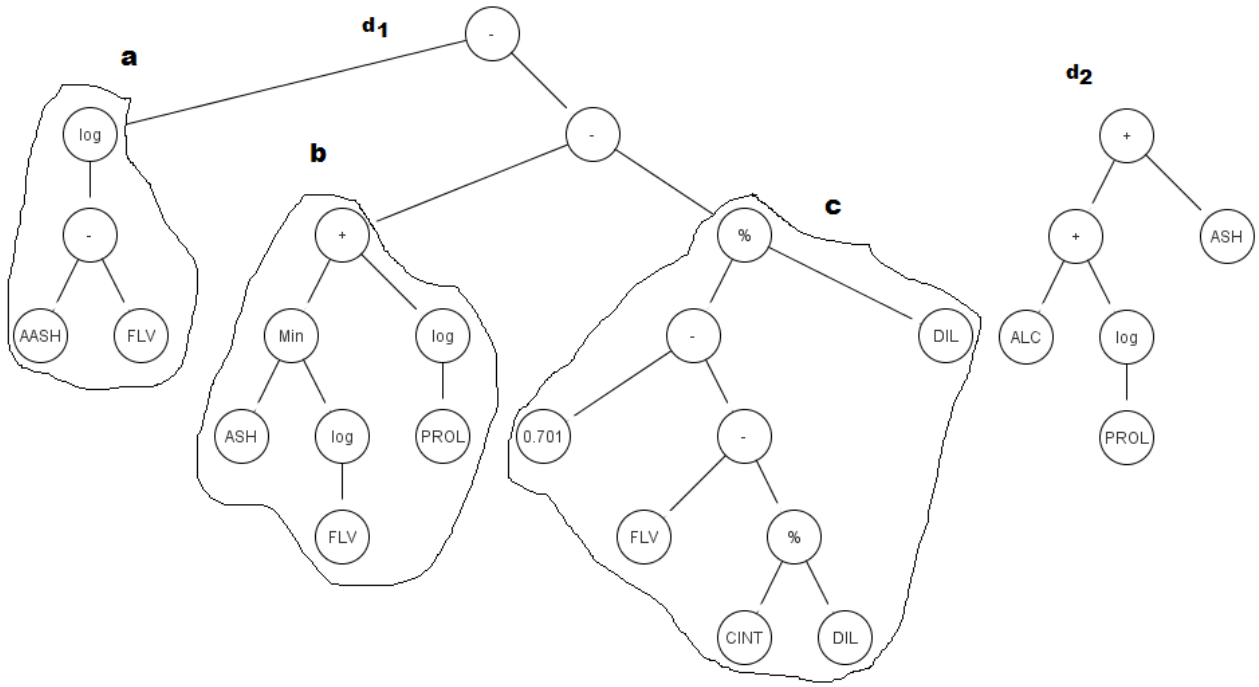


Figure 6.12: MOG3P expression trees for scatterplot (a) with a total size of 28

The pairs of expressions (from the seeded MOG3P runs) generating the visualizations in plots (b) and (c) are given below:

$$
\begin{aligned}
d_1 =& -0.2291 + HUE + -0.0035 * PROL + 0.117 * DIL + \\
& -0.7251 * ALC + 0.1396 * AASH + 0.0040 * MG + -0.2198 * CINT + \\
& -0.3402 * MACD + 2.9016 * NFLV + -2.2311 * ASH \\
d_2 =& DIL + FLV + -0.3639 * CINT
\end{aligned}
$$

$$
\begin{aligned}
d_1 =& FLV \\
d_2 =& 0.2967 * PROA + 1.4508 * NFLV + 0.1396 * AASH + 1.7069 * HUE + \\
& - 0.5843 * MACD + -0.0035 * PROL + 0.0040 * MG + 0.3984 * FLV + -0.4396 * CINT + \\
& - 0.7251 * ALC + -2.2311 * ASH
\end{aligned}
$$

Even though both sets of transformation functions bear a striking resemblance to the seeded expressions, they are much smaller in size (46 and 44 versus the original 102) while the quality of the generated visualizations are not affected in any way. However, also due to the resemblance, the percentage of feature elimination is low. In both cases, 11 out of the 13 attributes were used in generating the visualizations. Overall, MOG3P with seeding seems to be a useful direction in hybridizing the standard methods and evolutionary techniques for data visualization.

6.5 Analysis of Over-fitting in MOG3P

Over-fitting is a well-known problem in machine learning which deals with identifying and preventing lack of generalization power in training of learning algorithms. A formal definition of over-fitting is given by Mitchell in [104]:

Definition 3. Given a hypothesis space H, a hypothesis h $\in$ H is said to overfit the training data if there exists some alternative hypothesis h$'$ $\in H$, such that h has smaller error than h$'$ over the training examples, but h$'$ has a smaller error than h over the entire distribution of instances.

Over-fitting in GP has been studied by a number of researchers. Vanneschi et al. considers Mitchell's definition given above and proposes a way to measure 'the amount of over-fitting' of a GP based system in [152]. The authors acknowledge that a proper measure for over-fitting is not possible since it would require exploration of the while hypothesis space. Hence, they define a measure based on the relationship between the performance on the training data and the performance on unseen data (test set). The pseudocode for computation of this measure is given in algorithm 15. The following figure summarizes the process.

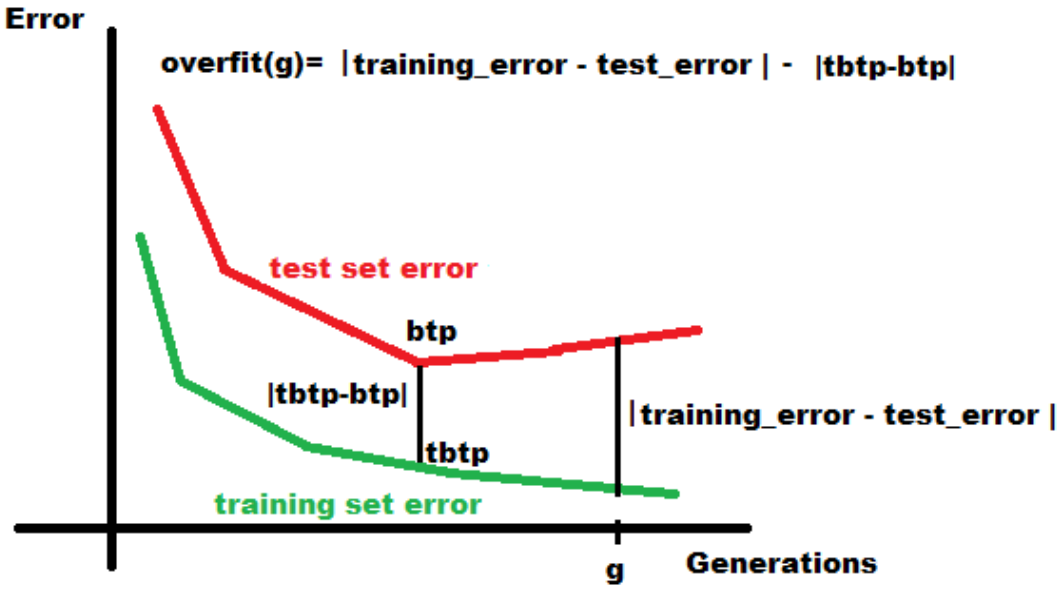


Figure 6.13: Computation of the over-fitting measure

In a GP based setting, as the generations advance, the algorithm has a potential to tune more and more to the training data to a degree that it loses the power to generalize to the unseen (test) data. This is the typical manifestation of over-fitting, an example of which can be seen in the figure presented above. The over-fit measure devised by Vanneschi et al. aims to measure the severity of over-fitting at each generation.

In summary, at each generation, this measure identifies of the best solution with the smallest training error and also records the corresponding test set error as the best values encountered so far. If the the best test error is less than the best training error than there is no over-fitting. Likewise, if the test set error is smaller than the current best test set error, there is no over-fitting. Over-fitting only occurs when the test set error starts to get larger than both the training set error and the smallest test error encountered so far. This measure assumes an 'elitist' GP implementation, where the best training set error never increases throughout the evolutionary process.

Algorithm 15: Over-fitting measure proposed in [152]

```
//Smaller training and test set errors are considered better solutions
overfit(0) = 0;
//btp: best test set performance
//tbtp: training set performance of the individual with the best test set performance
//training_error(g): smallest training error in the population at generation g
//test_error(g) : test error corresponding to the individual with the smallest training error
btp=test_error(0);
tbtp = training_error(0);
foreach generation > 0 do
    if training_error(g) > test_error(g) then
        overfit(g) = 0;
    else
        if (test_error(g) < btp) then
            overfit(g)=0 ;
            btp = test_error(g);
            tbtp= training_error(g);
        else
            overfit(g) = |training_error(g) - test_error(g)| - |tbtp − btp|;
        end
    end
end
```

Vanneschi et al. investigate the relationship between the functional complexity of the expressions and over-fitting in [152]. The authors state that effectively eliminating bloat does not necessarily prevent over-fitting and they attribute the observed over-fitting to functional complexity of the expressions. The authors define the functional complexity of an expression in terms of a crude estimation of curvature that is computed separately in each dimension. On a number of symbolic regression problems, the authors confirm that over-fitting still occurs even if bloat is completely eliminated, however the relationship between over-fitting and the curvature based functional complexity was inconclusive. Trujillo et al. investigate two functional complexity methods (slope-based and regularity-based) in [147]. The authors also report almost no correlation between these functional complexity measures and over-fitting on the same regression problems as in [152]. In both papers, the amount of over-fitting is computed using the algorithm given above. A similar argument on the relationship between functional complexity and generalization power is given by Vladislavleva et al. in [158] where the authors define a measure of nonlinearity of an expression in terms of Chebyshev polynomial approximation and use this measure along with accuracy and tree size objectives. The authors report improved generalization and compact expressions due to use of the measure of nonlinearity on a real-life symbolic regression problem.

The measure given above considers only the best-of-run individuals according to the training set error throughout the evolutionary process. However, there is also a whole population of individuals to

consider within each generation which constitutes a trade-off between training and test set performance. Some individuals will have better training set performance than others while lacking success in terms of test set performance (figure 6.14). Therefore, it would be beneficial for the algorithm to take advantage of this trade-off in assessing generalization power of the evolved models.

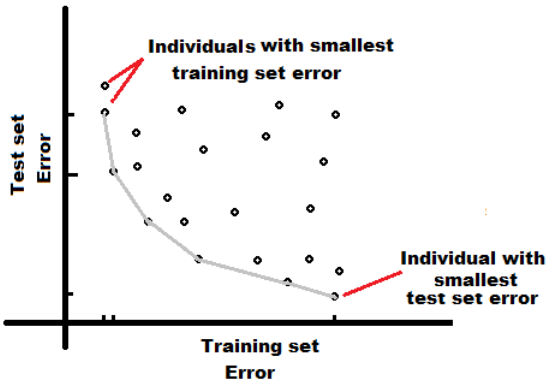


Figure 6.14: Training set - test set performance trade-off within a GP population

However, using the test set in any way to provide feedback to the training process is considered as the 'capital sin' in machine learning since the test set performance won't be a fair estimation of the generalization ability. In this case, the solution is to allocate a portion of the training set as the validation set and use the remaining of the training data as the training set.

The use of a separate validation set during the evolutionary process has been advocated in a few papers. Gagné et al. present a comparative study on the effects of using a separate validation set with or without explicit expression size control in [59]. The authors investigate the performance of GP-based classifiers on number of binary classification problems. At each generation, the set of best individuals with respect to training set performance and expression size are evaluated on the validation dataset. Then the best-of-run individual is selected as the one with smallest validation set error and ties are resolved in favor of smaller expressions. In this scheme, the validation set does not play a part in guiding the evolutionary process, since there is no feedback from validation phrase to the evolutionary process. The algorithm merely records the best-of-run individual throughout the run with respect to the experimental set up such as using a validation set with or without explicit parsimony control.

The authors report that using a validation set maintains accuracy while slightly reducing variance

of the test set results for the best-of-run solutions. They consider this reduction in variance as a sign of a little bit more stability in the results. Moreover, according to the authors, the use of validation set along with tree size control provides the reduction in variance of the test set results as well as the computational effort due to selection of more compact expressions.

Radtke et al. propose a global validation strategy for multi-objective evolutionary optimization based data classification in [120]. In this validation methodology, at each generation, the algorithm identifies a number of solutions that perform the best on the optimization data (the training dataset) and on the selection data (the validation dataset) as the best generalizing solutions. The solutions are kept in an auxiliary archive that is updated through the course of the evolutionary process. The evolutionary process is driven by the performance on the training set and complexity of the classifier that is evolved. At each generation, current best individuals are considered to be added on the archive based on their performance on the validation set. Therefore, the archive is guaranteed to contain the best individuals found so far that also offer good generalization to the independent dataset. The authors emphasize that, the validation process does not manipulate the evolutionary process in any way since the auxiliary archive is not exploited during the process to drive the optimization thus preventing the algorithm from over-fitting to the validation data.

Vanneschi and Gustafson also investigate the use of validation set during the evolutionary process in [153]. Additionally, the authors also aim to actively manipulate the population towards solutions that demonstrate more generalization power. This is achieved by flagging those individuals that are found to be over-fitting to the training dataset. These flagged individuals are called the 'repulsors' and in the parent selection stage, the probability that an individual is selected to participate in breeding is determined by its similarity to the repulsors. The authors define two genotype-based similarity measures: structural distance (edit-distance based) and subtree-crossover based similarity measure. Based on empirical evaluation on a real-life bioinformatics problem, the authors report that their method of preventing the over-fitting individuals and their likes from participating into the evolutionary process increases generalization power as measured on the test set as opposed to the standard GP implementation. They also

report that there is no statistical difference between using either similarity measure in this scheme. In this method, the validation set is used in driving the evolutionary process. The authors re-create the training set and validation partitions at fixed time intervals in order to help reduce the risk of over-fitting to the validation data.

The methods presented by Radtke et al. and Gagné et al. can be considered as passive approaches, in which the evolutionary process is monitored and good solutions are recorded as soon as they appear but no other action is taken in order to ensure that the search is directed towards similar solutions. On the other hand, Vanneschi and Gustafson's approach actively manipulates the population by attempting to eliminate the over-fitting individuals from the evolutionary process thus driving the search away from similar solutions.

6.5.1 Experiments

The first set of experiments aim to investigate how much over-fitting occurs during the MOG3P evolutionary process and if the choice of feature complexity method (tree size, expression complexity, weighted expression complexity, structural complexity and weighted structural complexity from chapter 3.3) would make any difference. Figure 6.15 show the results on the BUPA, PIMA and Wine datasets. The plots show the best training error encountered at each generation, test error for the MOG3P with the best training error, the amount of over-fit computed using the algorithm given above and the distribution of test set errors for the best-of-run solutions. Note that the test set errors are computed for generating these plots only. They were not used in any way to drive the process.

The result shows that over-fitting is observed at varying levels during the evolutionary process. The training error follows a declining pattern across generations which is the normal result of the evolutionary process trying to generate better solutions on the training data. The patterns for the test errors fluctuate widely for all experiments and all datasets. The amount of over-fit seems to be steadily increasing with ups and downs along the way. Based on these results, we found no evidence that using any feature complexity method makes a difference in terms of the amount of over-fitting.

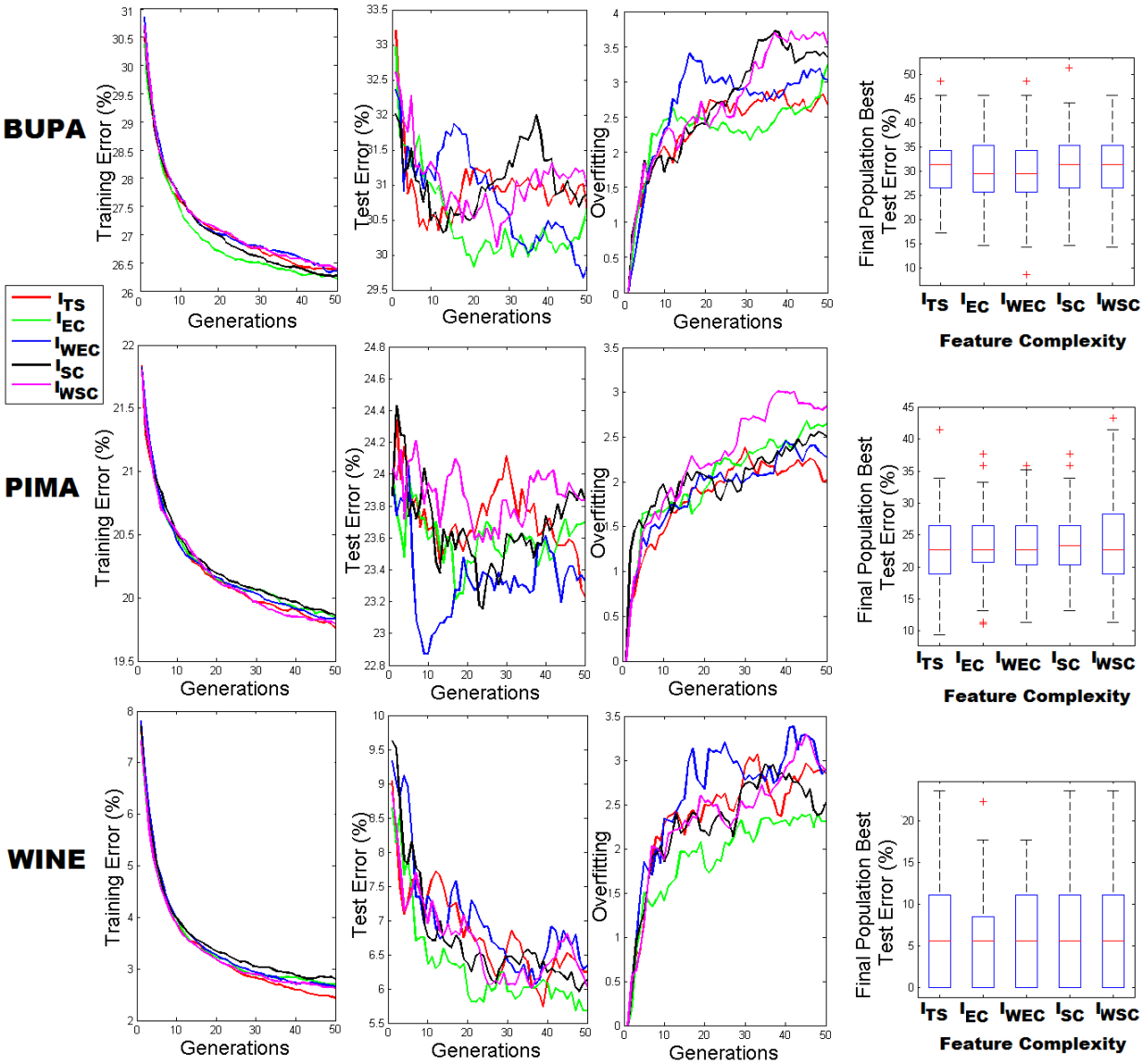


Figure 6.15: Analysis of over-fitting on BUPA, PIMA and WINE datasets with respect to different feature complexity measures

In the next set of experiments, we investigate if using two objectives (classifiability and feature complexity) and three objectives (classifiability, visual interpretability and feature complexity) would make any difference in terms of over-fitting. Figure 6.16 shows the comparative results. We found that adding visual interpretability had no significant negative or positive effect on the observed over-fitting.

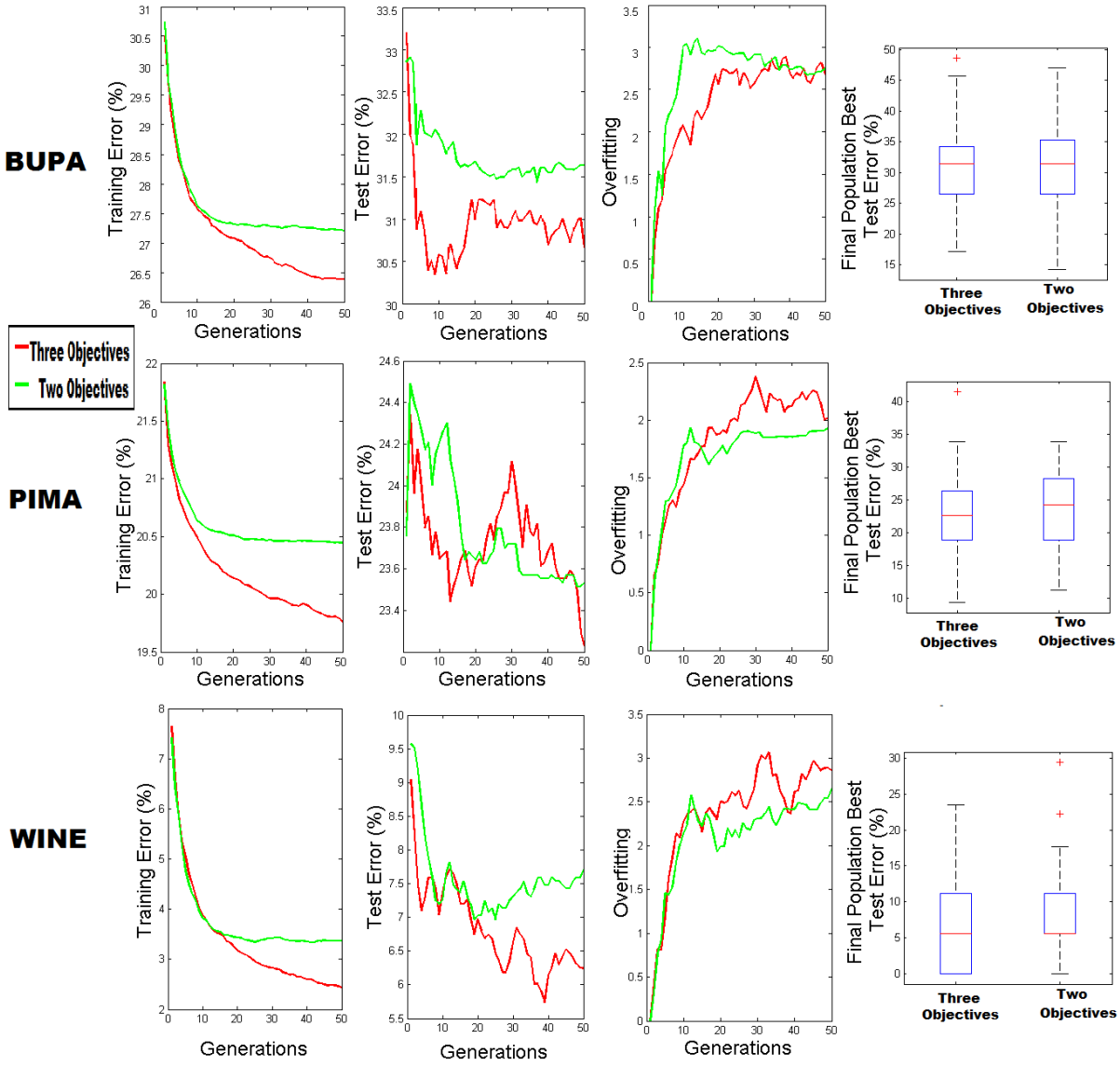


Figure 6.16: Analysis of over-fitting on BUPA, PIMA and WINE datasets with respect to two or three objectives (feature complexity: I_{TS})

Finally, we consider incorporating an early stopping strategy as a way to prevent over-fitting. Early stopping is a technique which was introduced in neural network training in order to detect when too much training starts to get harmful in terms of the generalization power [129]. Since it is not appropriate to use the test set to determine when to stop the evolutionary process, we further divide the training set into two parts in the following manner: we randomly select %67 of the data for training by keeping the class proportions the same as in the original training set and allocate the rest of the data as the

validation set. We continue to compute over-fitting using algorithm 15 but this time, we use the validation set instead of the test set. We also keep track of the over-fitting on the test set in order to be able to plot the results and compare validation set over-fitting to test set over-fitting.

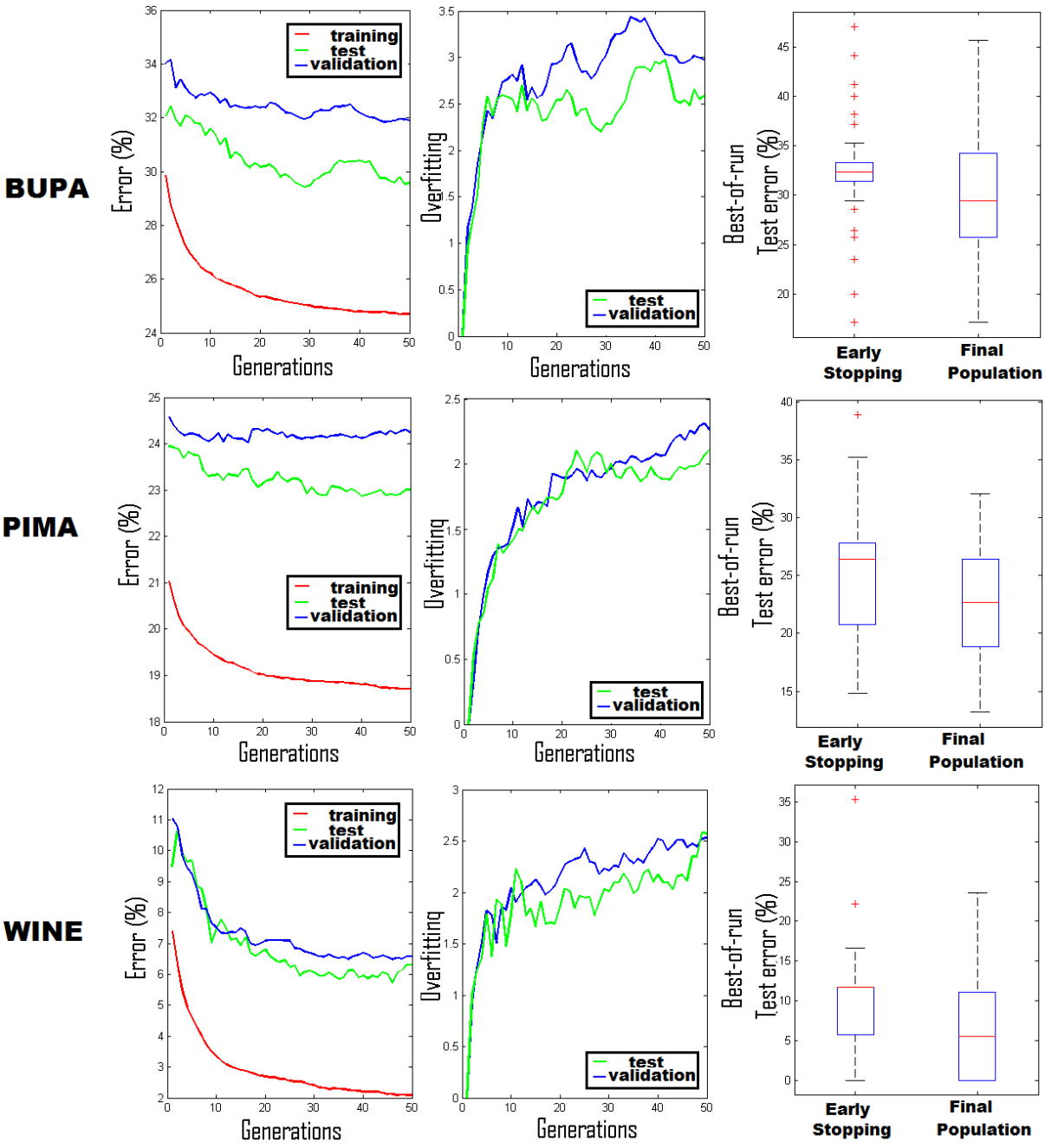


Figure 6.17: Results of early stopping on BUPA, PIMA and WINE datasets (three objectives, feature complexity: I_{TS})

After each run is completed, we find the generation with the smallest over-fit value and lowest training error and ignore the rest of the generations. We note that this is not a true stopping strategy, we still let the algorithm to run for the 50 generations as it was initially set. Figure 6.17 shows the results of employing this strategy in comparison to just using the MOG3P solutions from the final population. In all cases, there is no statistically significant difference between the test test errors of the selected solutions for all 100 runs. However, for the BUPA dataset, the test set errors were confined to a much smaller range, which indicates a more stable outcome than the test set results for the solutions from the final population. Results for the PIMA and Wine datasets do not demonstrate the same pattern.

In summary, the results of this section indicates that over-fitting is possible in MOG3P regardless of the use of different feature complexity measures. Including the visual interpretability measure has no negative or positive effect on generalization power. Incorporating a separate validation set within the MOG3P evolutionary process helps us compute a measure of over-fitting which can be used as an estimate of the generalization ability within the run. However, there is no guarantee that it would improve the results on the test set which is normal since the validation error is not a $\%100$ accurate estimation of how the model will perform on the test data.

6.6 Discussion

This chapter presented experiments in

- comparing single objective G3P to multi-objective implementation (MOG3P),
- incorporating known good solutions as *seeds* into the initial population,
- measurement and detection of over-fitting, early stopping as a way to avoid over-fitting in MOG3P

We also conducted various experiments using different population size, number of generations and initial tree depth parameters (Appendix B). In summary, increasing population size and generations increase performance (accuracy and visual interpretability) on the training data. However, the test set

performance does not necessarily improve significantly.

Through experiments we show that the multi-objective implementation requires much less experimentation in terms of identifying proper trade-offs between the multiple objectives. MOG3P generates a richer set of solutions compared to the G3P which are also superior in terms of Pareto dominance.

Through seeding, prior knowledge about the problem domain can be incorporated into the evolutionary process. In the case of MOG3P, seeding helps improve a data transformation function computed using a standard dimensionality reduction algorithm in terms of expression complexity.

Over-fitting is an important open problem in data classification. We incorporated a measure from the genetic programming literature to quantify the amount of over-fitting at each MOG3P generation. We then experimented with an early stopping criterion based on validation set performance. However, this was not found to significantly decrease over-fitting on the test set for the problems domains we studied.

Chapter 7

Conclusion

7.1 Summary and Contributions

The research that is presented in this thesis is at the intersection of the visual analytics and evolutionary computing fields. From the visual analytics perspective, we study interpretable dimensionality reduction for visual data classification. From the genetic programming perspective, we study the effect of single objective versus multi objective implementation, bloat prevention, detection and prevention of overfitting.

We itemize the contributions of this thesis as follows:

- *Evolutionary Computing based Automatic Search for Interpretable Dimensionality Reduction for Visual Data Classification*

  We introduced a genetic programming based automatic technique to create interpretable visual representations for higher dimensional labeled datasets for data classification. Instead of manually exploring the set of potentially useful visualizations, the algorithm provides the users with a number of good visualizations which are visually interpretable in terms of separability and compactness of each group on the view and semantically interpretable in terms of the complexity of the

visualization axes. The algorithm also identifies the most appropriate classifier for the generated views from a set of pre-specified classifiers. Since the complexity of the feature transformation functions are controlled by the algorithm, it tends to eliminate the unnecessary attributes that are not contributing to the quality of the extracted feature representation.

We show that the algorithm finds multiple good views on a number of benchmark data mining problems using simpler feature transformation functions compared to a number of standard techniques such as the PCA (principal components analysis), MDA (multiple discriminants analysis) and TPP (targeted projection pursuit).

- *A formal user study on two important aspects of interpretability for dimensionality reduction for visual data classification*. For visual interpretability, we study how much the automated measures of data visualization quality measures would match the human perception. We found that, although there is not a single measure capturing human intuition on judging 2D scatterplots, classification algorithms and a number of quality measures introduced in the visual analytics literature significantly correlate with human responses. Also cluster validation indices from the machine learning literature work well depending on the data characteristics.

  For interpretability of the visualization axes, we cast the problem as readability and comprehensibility of data transformation functions that can take any algebraic form. Based on our user study, we devised a measure of expression complexity in terms of structural characteristics of the syntax tree (size, depth, number and sizes of identifiable blocks). We utilize this measure of expression complexity within the G3P framework in order to help evolve human interpretable data transformation functions.

  Interpretability is an important concept in genetic programming. However, best to our knowledge, ours is the only formal user study on the interpretability of evolved feature transformations using GP.

- *A study on reproducibility and repeatability of data classification results using the cross-validation*

evaluation procedure. The evaluation protocol used in reporting of data classification results affect how the work should be interpreted or compared to similar techniques that are studying the same problem. Unfortunately, the practice varies in terms of how the cross-validation strategy is set up and how the significance of the results are reported using statistical tests. In our study, we investigated the effects of random partitioning on the outcome of the experiments. As it was pointed out by in machine learning literature, in order to ensure stable results, a large number of cross-validation iterations should be employed. However, due to computational constraints, this is not always possible. Therefore, we opted for repeatability through the use of a fixed set of random number seeds to create cross-validation folds so that we can keep the number of cross-validation iterations at 10. For GP based data mining experiments, the researcher also needs to decide how many runs to spare on the same input data in order to smooth out the effect of the variation in the initial population. Given that the total number of runs is practically limited, the trade off is either to vary the initial population or vary the input dataset (via more cross-validation iterations). Based on empirical analysis on a number of datasets, we decided to vary the input dataset thus more cross-validation iterations rather than sparing more runs on the same input data. We also adopted a corrected version of t-test for pair-wise comparison of experiment results. For the sake of comparability of the results across publications, we raised the issue of directly copying the performance results from other publications in order to compare one's results to other methods. Due to the differences in experimental procedure and the effects of random partitioning, there is a good chance that the results would not be fairly comparable. We make our cross-validation datasets and raw results available for anyone who would like to directly compare their techniques to our work [1].

- *Hybridizing the standard dimensionality reduction methods and evolutionary computing*

  The G3P/MOG3P does not enforce a specific algebraic form on the feature transformation functions. It can be any linear/non-linear function of the given set of original attributes. Even though

[1] See `http://web.cs.gc.cuny.edu/~iicke/g3p` for more information.

this provides flexibility, it also poses challenge since the space of possible feature transformation functions is vast, especially without any constraints on the expressions. Therefore, we also consider the best feature transformations generated by the standard methods such as the PCA, MDA or TPP as starting points for optimization via 'seeding' them into the initial population for MOG3P. During the evolutionary process, these solutions are either eliminated if they are not good enough or improved. For instance, in our experiments on the Wine dataset, we found that the MDA method generated a view with perfect separation and we incorporated the MDA generated feature transformation expression into MOG3P. The algorithm found simpler versions of these expressions without sacrificing the quality of the view.

7.2 Related Research Directions

In this section, we present various related research ideas that can either help further improve the G3P/MOG3P or can be useful applications of the algorithm.

7.2.1 Scalability: GPUs and Map-Reduce Framework

The major drawback of evolutionary computing method is the computational burden. The use of specialized hardware is becoming a common option to speed up the computation. Amongst these, the Graphical Processing Units (GPUs), have received increased attention by the evolutionary computing community. Initially, the GPUs were designed to carry out calculations in a massively parallel manner in order to create high quality graphics as fast as possible to support computer games. Later, the scientific community started to take advantage of the GPUs and nowadays, the most powerful GPUs are manifactured for scientific applications rather than computer games. In genetic programming, the most common use of the GPUs is to evaluate each individual's expression trees separately. This modification alone has been reported to provide major improvements in run time (see for example [124]). MOG3P can also benefit from GPUs in terms of scalability such as computing transformations of a set of individ-

ual data points in parallel. Another method to increase scalability for data intensive problems is through frameworks such as map-reduce [155].

7.2.2 Self Organizing MOG3P

Any GP system requires certain parameters to be tuned. Population size, number of generations, allowed depth limits for the initial trees and the set of functional symbols are major parameters to be set. The MOG3P employs user defined settings. Dynamic modification of the population size, stopping criteria and elimination of unnecessary functional symbols (furthermore elimination of subtrees via simplification) are important foreseeable improvements on MOG3P.

7.2.3 Beyond Scatterplots

In this dissertation, we focused on 2D scatterplots for data visualization and the visual quality measures were based on this form of visualization. Another form of visualization that is not confined to generating only two features is the parallel coordinates method (see figure 2.3 for an example) introduced by Inselberg which is also used for visual analysis of labeled datasets [71]. A number of visual quality measures for parallel coordinates in classification problems have been introduced in [143]. In MOG3P, it is possible to replace scatterplots with parallel coordinates along with the respective visual quality measures.

7.2.4 Classifier Domains of Competence

The relationship between the characteristics of the data and the classifiers has been an interesting research problem in machine learning. It has been largely argued that certain classifiers work better on datasets that possess certain characteristics which lead to the term ‘classifier domains of competence’ introduced by Kam Ho and Bernad-Mansilla in [77]. The authors define a number of measures to characterize a dataset and propose to model the domains of competence for each classifier in terms of the

defined data characteristics. This line of research has recently received attention from the evolutionary computing community. The great potential of the evolutionary methods has been exploited to generate synthetic datasets to analyze classifier behavior in [99]. The MOG3P creates large number of 2D feature sets that are evaluated using a set of classifiers. It is possible to characterize each of these feature sets in terms of the measurements proposed in [77] and study what types of feature transformations work better on certain datasets and classifiers based on analysis of the MOG3P evolutionary process.

7.2.5 Hybrid Supervised/Unsupervised Dimensionality Reduction using GP

In G3P/MOG3P, we incorporated supervised measures for dimensionality reduction which are based on the label information. The unsupervised methods are based on preservation of the distances of data points in the original space where the points that are closer to each other in the original space should appear close on the 2D visualization. However, defining and computing a distance measure in high dimensional spaces is difficult. The Euclidean distance in high dimensional spaces can be counter-intuitive because of the curse of dimensionality [20]. Alternative measures to Euclidean measure in the full space can be constructed by searching for a subset of dimensions that would make the distance calculation free of the effects of the curse of dimensionality ([6, 156]). The preservation of closeness can be implemented in MOG3P in parallel with a search for a proper distance measure in the original data space.

7.2.6 Interactive G3P/MOG3P

The G3P/MOG3P optimizes a predefined set of objective criteria and generates multiple models for user inspection. Namely, the user interaction is limited to the analysis of the best-of-run (final population) solutions. It is also possible to incorporate user interaction during the evolutionary process by visualizing a sample of individuals and ask for user feedback. The user's input can be used to eliminate certain individuals from the population and prevent similar individuals to be evolved again.

Appendix A

Source Code for the Corrected Repeated k-fold CV Test

```
function [h,p]=correpttest(x1,x2,iterations,folds,ntrain,ntest,alpha)
% [h]=correpttest(x1,x2,iterations,folds,ntrain,ntest,alpha)
% Corrected repeated k-fold cv test from Bouckaert and Frank,2004 paper
% titled:
% 'Evaluating the Replicability of Significance Tests for Comparing
% Learning Algorithms'
% x1--> vector of performance estimates for algorithm 1
% x2--> vector of performance estimates for algorithm 2
% NxK cross-validation --> N:iterations, K:Folds
% ntrain-->training set size
% ntest-->test set size
% alpha--> significance level for two-sided paired t-test
  nk=iterations*folds;

  %get the value of t-statistic at significance level alpha and
  %degree of freedom df
  tcv = tinv(1-(alpha/2),nk-1);
  tval= (abs(mean(x1)-mean(x2)) /
        sqrt( ( (1/nk)+(ntest/ntrain))*(var(x1-x2))));

  if tcv < tval
        h = 1;
  else
        h = 0;
  end;
   p = 2 * tcdf(-abs(tval), nk-1);
```

Appendix B

Effect of GP parameters on MOG3P

As in every evolutionary computing system, the performance of MOG3P also depends on the configuration parameters. In this section, we investigate the effect of population size, number of generations and the minimum and maximum tree depth settings for the tree initialization method. The configurations are summarized on table B.1.

Configuration	Generations	Population size/Archive size	Min-max (Ramped Half & Half Initialization)
C1	50	100/50	2-6
C2	50	100/50	2-10
C3	50	100/50	5-15
C4	50	500/250	2-10
C5	100	250/125	2-10
C6	100	1000/500	2-10

Table B.1: MOG3P configurations

First, we compare the effect of the tree depth settings for the initialization process. Tables B.2-B.4 show the results for population size 100, generations 50 and three GP tree initialization schemes. Figure B.2 shows the results at each generation. Based on the results, we infer that increasing the depth of initial GP trees did not provide significant increase in MOG3P performance on these problem domains. However, the average tree size is higher in the case of increased initial tree depth.

Measure	C1 (2-6)	C2 (2-10)	C3 (5-15)
Training Set Accuracy (%)	72.92 (1.48)	72.96 (1.66)	72.98 (1.59)
Test Set Accuracy (%)	68.30 (6.96)	69.05 (6.99)	69.37 (6.52)
Mean Tree Size	6.11 (2.98)	11.23 (13.58)	16.38 (28.90)
1-Median Vis.	0.52 (0.03)	0.52 (0.03)	0.52 (0.03)

Table B.2: Effect of Initialization parameters on BUPA dataset

Measure	C1 (2-6)	C2 (2-10)	C3 (5-15)
Training Set Accuracy (%)	79.67 (0.75)	79.68 (0.78)	79.72 (0.80)
Test Set Accuracy (%)	76.54 (4.95)	76.82 (5.88)	76.66 (4.75)
Mean Tree Size	7.02 (2.38)	10.79 (10.27)	13.10 (31.56)
1-Median Vis.	0.64 (0.01)	0.63 (0.01)	0.63 (0.01)

Table B.3: Effect of Initialization parameters on PIMA dataset

Measure	C1 (2-6)	C2 (2-10)	C3 (5-15)
Training Set Accuracy (%)	96.50 (1.2)	96.48 (1.16)	96.67 (1.30)
Test Set Accuracy (%)	93.99 (5.28)	93.03 (5.69)	93.12 (6.26)
Mean Tree Size	6.24 (2.73)	7.5 (4.78)	11.39 (15.25)
1-Median Vis.	0.87 (0.02)	0.88 (0.03)	0.88 (0.03)

Table B.4: Effect of Initialization parameters on Wine dataset

In the next set of experiments, we investigate the effect of populations size where the number of generations (50) and the initial tree depth range (2-10) are fixed. Figures B.5- B.7 show comparisons for population sizes 100, 250, 500 and 1000 across 50 populations. According to corrected t-test ($\alpha = 0.05$), population size 1000 returns significantly higher results on the performance on the training set (accuracy and visual interpretability) compared to 100 and 250, no significant improvement was found from 500 to 1000. On test set, none of the population sizes resulted in significant improvement.

Measure	C2 (100)	C4 (500)	C5 (250)	C6 (1000)
Training Set Accuracy (%)	72.96 (1.66)	75.17 (1.27)	74.34 (1.25)	75.69 (1.06)
Test Set Accuracy (%)	69.05 (6.99)	70.74 (7.04)	69.98 (7.16)	70.66 (6.49)
Mean Tree Size	11.23 (13.58)	8.66 (6.36)	7.70 (5.63)	8.27 (2.97)
1-Median Vis.	0.52 (0.03)	0.56 (0.02)	0.55 (0.02)	0.57 (0.01)

Table B.5: Effect of population size on BUPA dataset (50 iterations)

Measure	C2 (100)	C4 (500)	C5 (250)	C6 (1000)
Training Set Accuracy (%)	79.68 (0.78)	80.87 (0.75)	80.51 (0.75)	81.21 (0.68)
Test Set Accuracy (%)	76.82 (5.88)	76.79 (5.26)	76.45 (5.29)	76.99 (5.08)
Mean Tree Size	10.79 (10.27)	9.33 (4.22)	9.05 (4.31)	10.54 (5.98)
1-Median Vis.	0.63 (0.12)	0.65 (0.08)	0.65 (0.08)	0.66 (0.09)

Table B.6: Effect of population size on PIMA dataset (50 iterations)

Measure	C2 (100)	C4 (500)	C5 (250)	C6 (1000)
Training Set Accuracy (%)	96.48 (1.16)	98.32 (0.81)	97.79 (0.95)	98.70 (0.72)
Test Set Accuracy (%)	93.03 (5.69)	95.38 (5.47)	94.33 (4.68)	95.82 (4.61)
Mean Tree Size	7.50 (4.78)	8.28 (4.92)	8.33 (3.88)	7.72 (2.02)
1-Median Vis.	0.88 (0.03)	0.91 (0.02)	0.91 (0.01)	0.93 (0.02)

Table B.7: Effect of population size on WINE dataset (50 iterations)

The final set of experiments investigate the effect of running MOG3P for more generations. Figure B.3 shows the effect of running MOG3P for 100 generations with population sizes 250 (C5) and 1000 (C6). On all three problem domains, increasing the number of generations helps increase performance on training data (accuracy and visual interpretability).

Figure B.4 shows the best of run performance indicators on all three problem domains. For larger population size the improvement is always statistically significant between 50 generations and 100 generations. However, we found no statistically significant differences in test set accuracy (according to the corrected t-test). Most significant difference on training set is achieved when both the population size and the generations are increased together. As for test set performance, the only noticeable difference is on the Wine dataset. This is due to the fact that this dataset can indeed be clearly separated, therefore a MOG3P solution with almost perfect training set performance will be likely to perform well on the test set as well. The other two datasets are known to be difficult classification problems due to overlap and possible outliers in the data, and best to our knowledge, there has not be any perfectly separating data transformations reported for these problems.

In summary, population size, generations and allowed initial tree depth have effects on the outcome of the MOG3P. For the problem domains we studied here, increasing the initial tree depth increases the size of the final solutions but no significant increase in performance. Increasing population size helps increase performance, as well as running MOG3P for longer generations. Increasing both the population size and number of generations always increases performance. However, the increase in performance is on the training set, when the solution is applied to the test set the improvement was not found to be significant in either of the problem domains studied here.

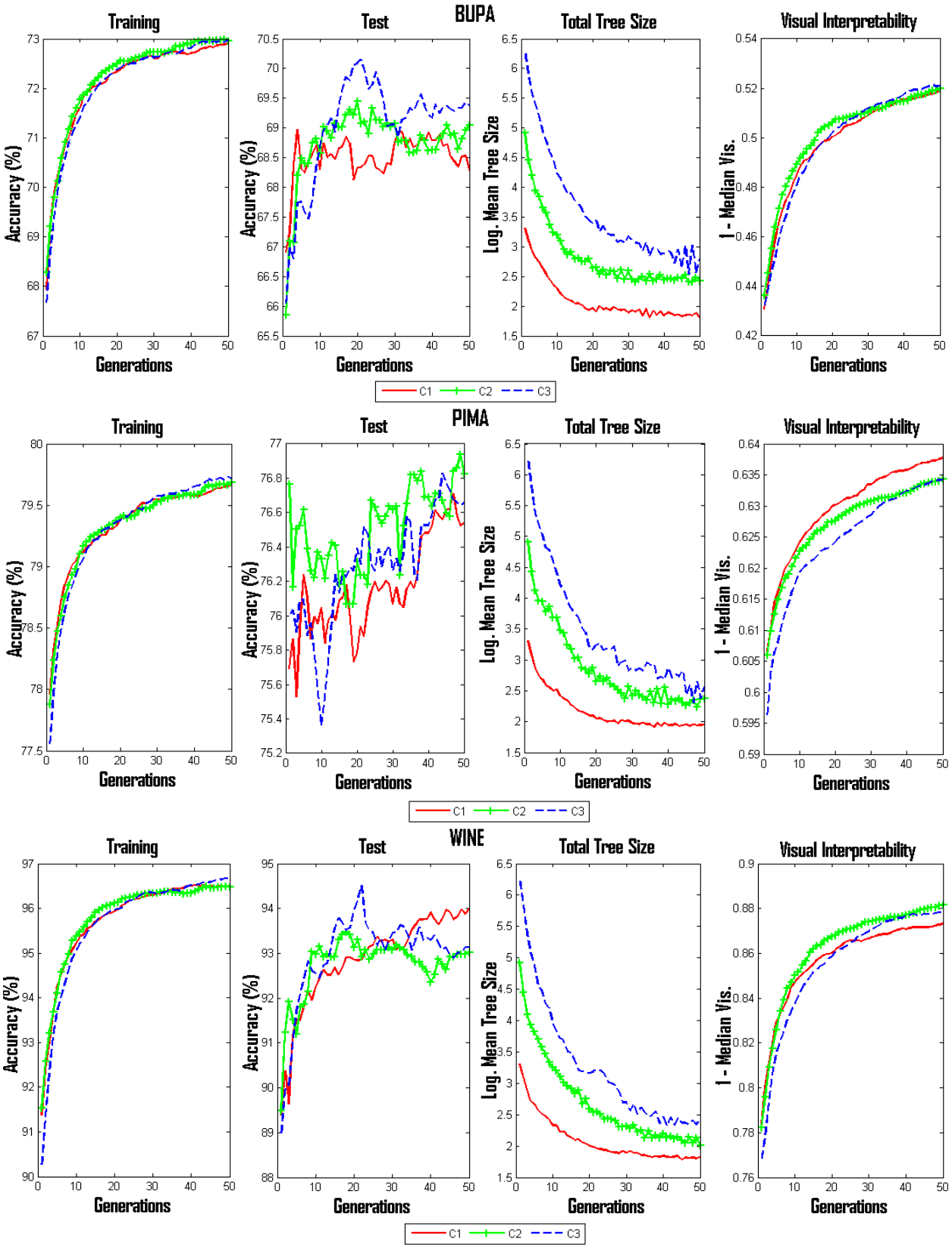


Figure B.1: Comparison of MOG3P initialization parameters

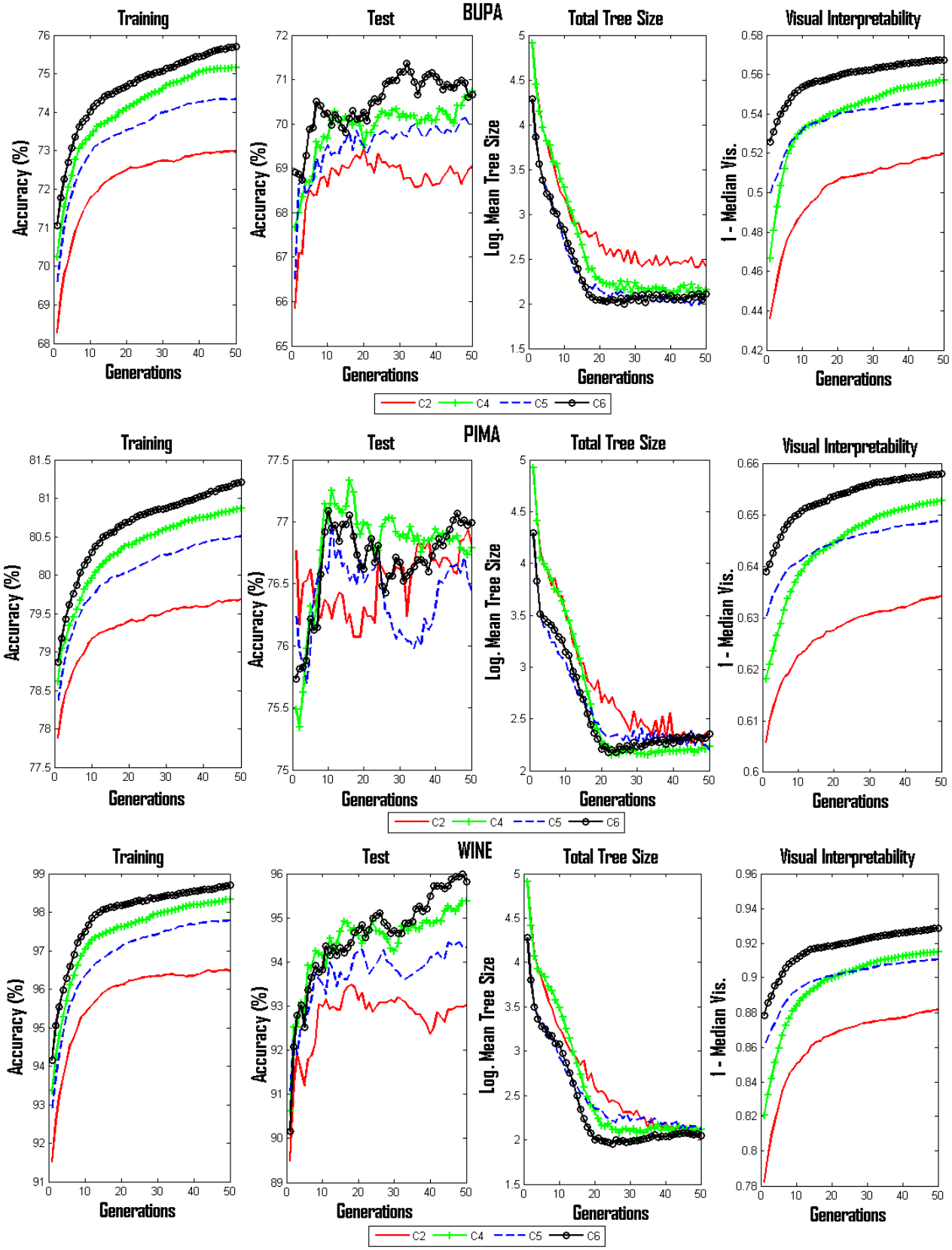


Figure B.2: Comparison of MOG3P population size (50 iterations)

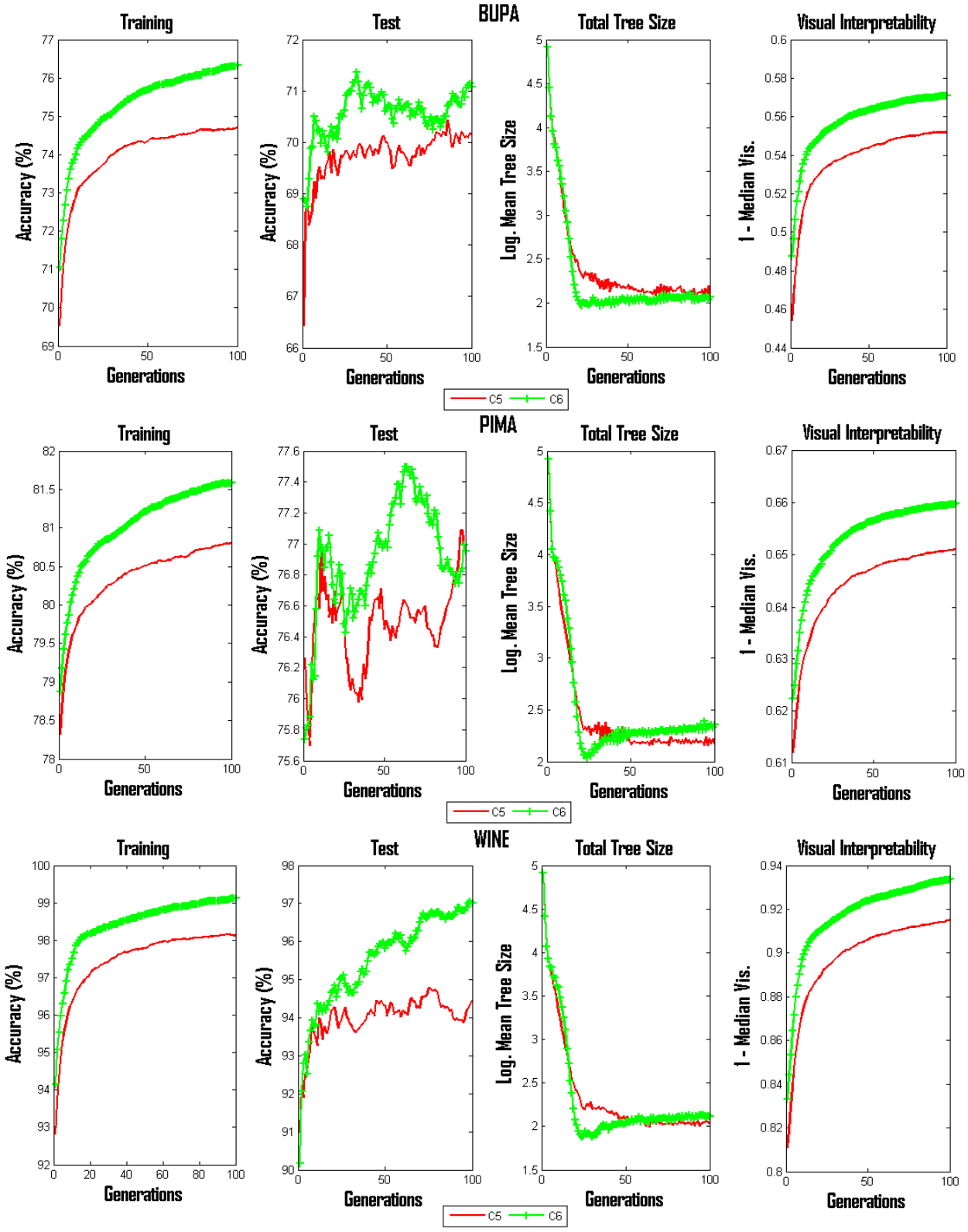


Figure B.3: Comparison of MOG3P generations (100 iterations)

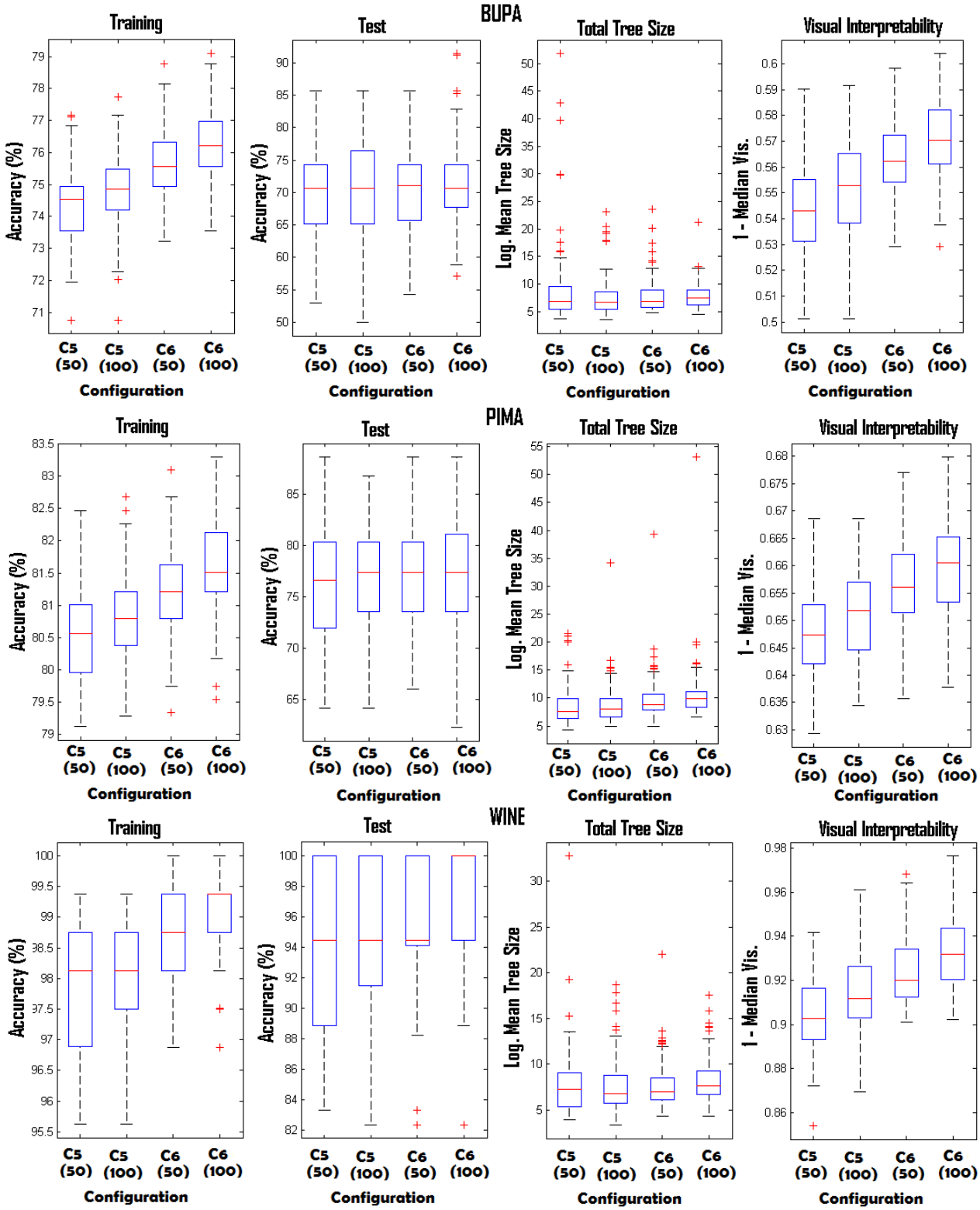


Figure B.4: Comparison of MOG3P generations (50, 100 iterations)

Bibliography

[1] Data modeler. http://evolved-analytics.com/.

[2] Italian olive oils dataset. http://www.ggobi.org/book/data/olive.csv.

[3] Uci machine learning data repository. `www.ics.uci.edu/~mlearn`.

[4] Weka machine learning toolkit. `http://www.cs.waikato.ac.nz/~ml/weka/`.

[5] D. Achlioptas. Database-friendly random projections. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '01, pages 274–281, New York, NY, USA, 2001. ACM.

[6] C. Aggarwal, A. Hinneburg, and D. Keim. On the surprising behavior of distance metrics in high dimensional space. In J. Van den Bussche and V. Vianu, editors, *Database Theory ICDT 2001*, volume 1973 of *Lecture Notes in Computer Science*, pages 420–434. Springer Berlin / Heidelberg, 2001.

[7] E. Alfaro-Cid, J. J. Merelo, F. F. de Vega, A. I. Esparcia-Alczar, and K. Sharman. Bloat control operators and diversity in genetic programming: A comparative study. *Evolutionary Computation*, 18(2):305–332, 2010.

[8] E. Alpaydin. Combined 5x2cv f test for comparing supervised classification learning algorithms. *Neural Computation*, 11:1885–1892, 1999.

[9] D. Andrews. Plots of high dimensional data. *Biometrics*, 28:125–136, 1972.

[10] M. Ankerst, S. Berchtold, and D. A. Keim. Similarity clustering of dimensions for an enhanced visualization of multidimensional data. *infovis*, 00:52, 1998.

[11] L. Anthony, J. Yang, and K. R. Koedinger. Evaluation of multimodal input for entering mathematical equations on the computer. In *CHI '05 extended abstracts on Human factors in computing systems*, CHI EA '05, pages 1184–1187, New York, NY, USA, 2005. ACM.

[12] A. Arcuri, D. R. White, J. Clark, and X. Yao. Multi-objective improvement of software using co-evolution and smart seeding. In *Proceedings of the 7th International Conference on Simulated Evolution and Learning*, SEAL '08, pages 61–70, Berlin, Heidelberg, 2008. Springer-Verlag.

[13] A. Awde, Y. Bellik, and C. Tadj. Complexity of mathematical expressions in adaptive multimodal multimedia system ensuring access to mathematics for visually impaired users. *International Journal of Computer and Information Science and Engineering*, 2:103–115, 2008.

[14] K. Badran and P. I. Rockett. The influence of mutation on population dynamics in multiobjective genetic programming. *Genetic Programming and Evolvable Machines*, 11:5–33, March 2010.

[15] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 14–21, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc.

[16] L. Beadle and C. G. Johnson. Semantically driven crossover in genetic programming. pages 111–116, 2008.

[17] L. Beadle and C. G. Johnson. Semantically driven mutation in genetic programming. pages 1336–1342, 2009.

[18] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:711–720, 1997.

[19] Y. Bernstein, X. Li, V. Ciesielski, and A. Song. Multiobjective parsimony enforcement for superior generalisation performance. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, pages 83–89, Portland, Oregon, 20-23 June 2004. IEEE Press.

[20] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbor meaningful? In C. Beeri and P. Buneman, editors, *Database Theory ICDT99*, volume 1540 of *Lecture Notes in Computer Science*, pages 217–235. Springer Berlin / Heidelberg, 1999.

[21] S. Bhattacharyya. Evolutionary algorithms in data mining: Multi-objective performance modeling for direct marketing. In *In Proc. 6th ACM SIGKDD Intl Conf. on Knowledge Discovery & Data Mining (KDD2000)*, pages 465–473. AAAI Press, 2000.

[22] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250, New York, NY, USA, 2001. ACM.

[23] S. Bleuler, M. Brack, L. Thiele, and E. Zitzler. Multiobjective genetic programming: Reducing bloat using SPEA2. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 536–543, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27-30 May 2001. IEEE Press.

[24] C. C. Bojarczuk, H. S. Lopes, A. A. Freitas, and E. L. Michalkiewicz. A constrained-syntax genetic programming system for discovering classification rules: application to medical data sets. *Artificial Intelligence in Medicine*, pages 27–48, 2004.

[25] R. J. Bolton and W. J. Krzanowski. Projection pursuit clustering for exploratory data analysis. *Journal of Computational and Graphical Statistics*, 12(1):121–142, 2003.

[26] J. C. Bongard. A probabilistic functional crossover operator for genetic programming. pages 925–932, 2010.

[27] M. C. J. Bot. Feature extraction for the k-nearest neighbour classifier with genetic programming. In *EuroGP '01: Proceedings of the 4th European Conference on Genetic Programming*, pages 256–267, London, UK, 2001. Springer-Verlag.

[28] R. R. Bouckaert and E. Frank. Evaluating the replicability of significance tests for comparing learning algorithms. In *PAKDD*, pages 3–12, 2004.

[29] P. S. Bradley, U. M. Fayyad, and O. L. Mangasarian. Mathematical programming for data mining: Formulations and challenges. *INFORMS J. on Computing*, 11(3):217–238, 1999.

[30] C. J. C. Burges. Geometric methods for feature extraction and dimensional reduction. pages 59–92, 2005.

[31] E. K. Burke, S. Gustafson, and G. Kendall. Diversity in genetic programming: an analysis of measures and correlation with fitness. *IEEE Trans. Evolutionary Computation*, 8(1):47–62, 2004.

[32] E. K. Burke, S. Gustafson, G. Kendall, N. Krasnogor, and E. K. B. S. Gustafson. Is increased diversity in genetic programming beneficial? an analysis of lineage selection. In *Congress on Evolutionary Computation*, pages 1398–1405. IEEE Press, 2003.

[33] F. Camastra and A. Vinciarelli. *Feature Extraction Methods and Manifold Learning Methods*, chapter 11, pages 305–341. Springer London, 2008.

[34] L. Castro. *Fundamentals of natural computing: basic concepts, algorithms, and applications*. Chapman & Hall/CRC computer and information science series. Chapman & Hall/CRC, 2006.

[35] H. Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, 68(342):361–368, 1973.

[36] R. Chiong. *Nature-Inspired Algorithms for Optimisation*. Studies in Computational Intelligence. Springer, 2009.

[37] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[38] N. L. Cramer. A representation for the adaptive generation of simple sequential programs. *Proceedings of an International Conference on Genetic Algorithms and the Applications, Grefenstette, John J. (ed.), Carnegie Mellon University*, 1985.

[39] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, Apr. 1979.

[40] P. Day and A. K. Nandi. Evolution of superfeatures through genetic programming. *Expert Systems*, 28(2):167–184, 2011.

[41] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons, Chichester, 2001.

[42] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast elitist multi-objective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2000.

[43] K. DeJong. Parameter setting in eas: a 30 year perspective. pages 1–18, 2007.

[44] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, December 2006.

[45] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10:1895–1923, 1998.

[46] S. Dignum and R. Poli. Operator equalisation and bloat free gp. In *Proceedings of the 11th European conference on Genetic programming*, EuroGP'08, pages 110–121, Berlin, Heidelberg, 2008. Springer-Verlag.

[47] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

[48] J. C. Dunn. Well separated clusters and optimal fuzzy-partitions. *Journal of Cybernetics*, 4:95–104, 1974.

[49] A. E. *Introduction to Machine Learning*. The MIT press, 2004.

[50] A. Eiben and J. Smith. *Introduction to evolutionary computing*. Natural computing series. Springer, 2003.

[51] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Trans. Evolutionary Computation*, 3(2):124–141, 1999.

[52] A. Ekárt and S. Z. Németh. Selection based on the pareto nondomination criterion for controlling code growth in genetic programming. *Genetic Programming and Evolvable Machines*, 2:61–73, March 2001.

[53] C. Estèbanez, R. Aler, and J. Valls. A method based on genetic programming for improving the quality of datasets in classification problems. *International Journal of Computer Science and Applications*, 4:69–80, 2007.

[54] J. Faith, R. Mintram, and M. Angelova. Targeted projection pursuit for visualizing gene expression data classifications. *Bioinformatics*, 22(21):2667–2673, Nov 2006.

[55] G. Flake. *The computational beauty of nature: computer explorations of fractals, chaos, complex systems, and adaptation*. Bradford Books. MIT Press, 2000.

[56] A. A. Freitas. A critical review of multi-objective optimization in data mining: a position paper. *SIGKDD Explorations*, 6(2):77–86, 2004.

[57] A. A. Freitas. A review of evolutionary algorithms for data mining. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 371–400. Springer US, 2010.

[58] J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *Computers, IEEE Transactions on*, C-23(9):881–890, 1974.

[59] C. Gagné, M. Schoenauer, M. Parizeau, and M. Tomassini. Genetic programming, validation sets, and parsimony pressure. In *EuroGP*, pages 109–120, 2006.

[60] A. Ghosh, S. Dehuri, and S. Ghosh, editors. *Multi-Objective Evolutionary Algorithms for Knowledge Discovery from Databases*, volume 98 of *Studies in Computational Intelligence*. Springer, 2008.

[61] H. Guo and A. K. Nandi. Breast cancer diagnosis using genetic programming generated feature. *Pattern Recogn.*, 39(5):980–987, 2006.

[62] S. Gustafson. *An Analysis of Diversity in Genetic Programming*. PhD thesis, School of Computer Science and Information Technology, University of Nottingham, Nottingham, England, Feb. 2004.

[63] S. Gustafson, E. K. Burke, and N. Krasnogor. On improving genetic programming for symbolic regression. In D. Corne, Z. Michalewicz, M. Dorigo, G. Eiben, D. Fogel, C. Fonseca, G. Greenwood, T. K. Chen, G. Raidl, A. Zalzala, S. Lucas, B. Paechter, J. Willies, J. J. M. Guervos, E. Eberbach, B. McKay, A. Channon, A. Tiwari, L. G. Volkert, D. Ashlock, and M. Schoenauer, editors, *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 912–919, Edinburgh, UK, 2-5 Sept. 2005. IEEE Press.

[64] I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors. *Feature Extraction, Foundations and Applications*. Series Studies in Fuzziness and Soft Computing, Physica-Verlag, Springer, 2006.

[65] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. Cluster validity methods: part i. *SIGMOD Rec.*, 31:40–45, June 2002.

[66] J. Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[67] G. S. Hornby. Alps: the age-layered population structure for reducing the problem of premature convergence. pages 815–822, 2006.

[68] L. Hubert and J. Schultz. Quadratic assignment as a general data-analysis strategy. *British Journal of Mathematical and Statistical Psychologie*, 29:190–241, 1976.

[69] I. Icke and A. Rosenberg. Multi-objective genetic programming for visual analytics. In S. Silva, J. A. Foster, M. Nicolau, M. Giacobini, and P. Machado, editors, *Proceedings of the 14th European Conference on Genetic Programming, EuroGP 2011*, volume 6621 of *LNCS*, pages 323–334, Turin, Italy, 27-29 Apr. 2011. Springer Verlag.

[70] J. H. Imada and B. J. Ross. Using feature-based fitness evaluation in symbolic regression with added noise. In *Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, GECCO '08, pages 2153–2158, New York, NY, USA, 2008. ACM.

[71] A. Inselberg and T. Avidan. Classification and visualization for high-dimensional data. In *KDD'00*, pages 370–374, 2000.

[72] A. Inselberg and B. Dimsdale. Parallel coordinates: a tool for visualizing multi-dimensional geometry. In *VIS '90: Proceedings of the 1st conference on Visualization '90*, pages 361–378, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[73] Y. Jin, editor. *Multi-Objective Machine Learning*, volume 16 of *Studies in Computational Intelligence*. Springer, 2006.

[74] C. G. Johnson. Genetic programming crossover: Does it cross over? pages 97–108, 2009.

[75] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[76] E. D. D. Jong and J. B. Pollack. Multi-objective methods for tree size control. *Genetic Programming and Evolvable Machines*, 4(3):211–233, Sept. 2003.

[77] T. Kam Ho and E. Bernad-Mansilla. Classifier domains of competence in data complexity space. In M. Basu and T. K. Ho, editors, *Data Complexity in Pattern Recognition*, Advanced Information and Knowledge Processing, pages 135–152. Springer London, 2006.

[78] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. Visual analytics: Definition, process, and challenges. In *Information Visualization*, pages 154–175. Springer, 2008.

[79] D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, 2002.

[80] T. L. Khalid El-Arini, Andrew W. Moore. Autonomous visualization. In *European Conference on Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2006)*, Berlin, Germany, September 2006.

[81] K. Kira and L. A. Rendell. A practical approach to feature selection. In *Proceedings of the Ninth International Workshop on Machine Learning*, ML '92, pages 249–256, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[82] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2*, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[83] J. R. Koza. *Genetic programming - on the programming of computers by means of natural selection*. Complex adaptive systems. MIT Press, 1992.

[84] K. Krawiec. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genetic Programming and Evolvable Machines*, 3(4):329–343, 2002.

[85] I. Kushchu. Genetic programming and evolutionary generalization. *IEEE Transactions on Evolutionary Computation*, 6(5):431–442, 2002.

[86] W. B. Langdon. The evolution of size in variable length representations. In *International Conference on Evolutionary Computation*, 1998.

[87] W. B. Langdon. Size fair and homologous tree genetic programming crossovers. 1999.

[88] W. B. Langdon and P. Nordin. Seeding genetic programming populations. In *Proceedings of the European Conference on Genetic Programming*, pages 304–315, London, UK, 2000. Springer-Verlag.

[89] W. B. Langdon and R. Poli. *Foundations of genetic programming*. Springer-Verlag New York, Inc., New York, NY, USA, 2002.

[90] G. Leban, B. Zupan, G. Vidmar, and I. Bratko. Vizrank: Data visualization guided by machine learning. *Data Mining and Knowledge Discovery*, 13:119–136, 2006. 10.1007/s10618-005-0031-5.

[91] E.-K. Lee, D. Cook, S. Klinke, and T. Lumley. Projection pursuit for exploratory supervised classification. *Journal of Computational and Graphical Statistics*, 14(4):831–846, December 2005.

[92] J.-Y. Lin, H.-R. Ke, B.-C. Chien, and W.-P. Yang. Classifier design with feature selection and feature extraction using layered genetic programming. *Expert Syst. Appl.*, 34:1384–1393, February 2008.

[93] J. Luengo, S. García, and F. Herrera. A study on the use of statistical tests for experimentation with neural networks. In *Computational and Ambient Intelligence, 9th International Work-Conference on Artificial Neural Networks,IWANN*, pages 72–79, 2007.

[94] S. Luke. Ecj 20 a java-based evolutionary computation research system, `http://cs.gmu.edu/~eclab/projects/ecj/`.

[95] S. Luke. Code growth is not caused by introns. In *In Whitley, D. (Ed.), Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference (pp. 228 235). Las Vegas*, pages 228–235. Morgan Kaufmann, 2000.

[96] S. Luke. *Essentials of Metaheuristics*. Lulu, 2009. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

[97] S. Luke and L. Panait. Lexicographic parsimony pressure. In *Genetic and Evolutionary Computation Conference*, pages 829–836, 2002.

[98] S. Luke and L. Panait. A comparison of bloat control methods for genetic programming. *Evolutionary Computation*, 14(3):309–344, 2006.

[99] N. Macià, A. Orriols-Puig, and E. Bernadó-Mansilla. Genetic-based synthetic data sets for the analysis of classifiers behavior. In *HIS*, pages 507–512, 2008.

[100] H. Majeed and C. Ryan. Using context-aware crossover to improve the performance of gp. pages 847–854, 2006.

[101] H. Majeed and C. Ryan. Context-aware mutation: a modular, context aware mutation operator for genetic programming. pages 1651–1658, 2007.

[102] J. Mao and A. K. Jain. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transactions on Neural Networks*, 6:296–317, 1995.

[103] T. McConaghy and G. Gielen. Canonical form functions as a simple means for genetic programming to evolve human-interpretable functions. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, GECCO '06, pages 855–862, New York, NY, USA, 2006. ACM.

[104] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[105] S. C. Morton. *Interpretable Projection Pursuit*. PhD thesis, Stanford University, 1989.

[106] M. Muharram and G. D. Smith. Evolutionary constructive induction. *IEEE Trans. on Knowl. and Data Eng.*, 17(11):1518–1528, 2005.

[107] D. Muni, N. Pal, and J. Das. Genetic programming for simultaneous feature selection and classifier design. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(1):106–117, February 2006.

[108] C. Nadeau and Y. Bengio. Inference for the generalization error. *Machine Learning*, 52:239–281, September 2003.

[109] K. Neshatian and M. Zhang. Genetic programming and class-wise orthogonal transformation for dimension reduction in classification problems. In *EuroGP'08: Proceedings of the 11th European conference on Genetic programming*, pages 242–253, Berlin, Heidelberg, 2008. Springer-Verlag.

[110] M. ONeill, L. Vanneschi, S. Gustafson, and W. Banzhaf. Open issues in genetic programming. *Genetic Programming and Evolvable Machines*, 11:339–363, 2010.

[111] F. E. B. Otero, M. M. S. Silva, A. A. Freitas, and J. C. Nievola. Genetic programming for attribute construction in data mining. In *Genetic Programming, Proceedings of EuroGP2003, volume 2610 of LNCS*, pages 384–393. Springer-Verlag, 2003.

[112] G. L. Pappa and A. A. Freitas. *Automatically Evolving Data Mining Algorithms*, volume XIII of *Natural Computing Series*. Springer, 2010.

[113] V. Pareto. *Manual of Political Economy,*. London: Macmillan, 1972., 1906.

[114] D. Parrott, X. Li, and V. Ciesielski. Multi-objective techniques in genetic programming for evolving classifiers. In *Congress on Evolutionary Computation'05*, pages 1141–1148, 2005.

[115] R. Poli. General schema theory for genetic programming with subtree-swapping crossover. In *In Genetic Programming, Proceedings of EuroGP 2001, LNCS*, pages 18–20. Springer-Verlag, 2001.

[116] R. Poli. A simple but theoretically-motivated method to control bloat in genetic programming. In C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli, and E. Costa, editors, *Genetic Programming*, volume 2610 of *Lecture Notes in Computer Science*, pages 43–76. Springer Berlin / Heidelberg, 2003.

[117] R. Poli, W. B. Langdon, and N. F. McPhee. *A Field Guide to Genetic Programming*. Lulu Enterprises, UK Ltd, 2008.

[118] R. Poli, N. F. McPhee, and J. E. Rowe. Exact schema theory and markov chain models for genetic programming and variable-length genetic algorithms with homologous crossover. *Genetic Programming and Evolvable Machines*, pages 31–70, 2004.

[119] R. Poli, L. Vanneschi, W. Langdon, and N. McPhee. Theoretical results in genetic programming: the next ten years? *Genetic Programming and Evolvable Machines*, 11:285–320, 2010.

[120] P. V. W. Radtke, T. Wong, and R. Sabourin. An evaluation of over-fit control strategies for multi-objective evolutionary optimization. In *IJCNN'06*, pages 3327–3334, 2006.

[121] T. Raeder, T. R. Hoens, and N. V. Chawla. Consequences of variability in classifier performance estimates. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 421–430, Washington, DC, USA, 2010. IEEE Computer Society.

[122] E. Rendón, I. M. Abundez, C. Gutierrez, S. D. Zagal, A. Arizmendi, E. M. Quiroz, and H. E. Arzate. A comparison of internal and external cluster validation indexes. In *Proceedings of the 2011 American conference on applied mathematics and the 5th WSEAS international conference on Computer engineering and applications*, AMERICAN-MATH'11/CEA'11, pages 158–163, Stevens Point, Wisconsin, USA, 2011. World Scientific and Engineering Academy and Society (WSEAS).

[123] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, 1996.

[124] D. Robilliard, V. Marion, and C. Fonlupt. High performance genetic programming on gpu. In *Proceedings of the 2009 workshop on Bio-inspired algorithms for distributed systems*, BADS '09, pages 85–94, New York, NY, USA, 2009. ACM.

[125] E. Rodriguez-Martinez, J. Y. Goulermas, T. Mu, and J. F. Ralph. Automatic induction of projection pursuit indices. *IEEE Transactions on Neural Networks*, 21(8):1281–1295, 2010.

[126] J. Rosca. Entropy-driven adaptive representation. In *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 23–32. Morgan Kaufmann, 1995.

[127] A. Rosenberg. Autobi - a tool for automatic tobi annotation. In *INTERSPEECH 2010*, 2010.

[128] L. Sánchez, J. Otero, and J. Alcalá-Fdez. Assessing the differences in accuracy between gfss with bootstrap tests. In *EUSFLAT Conf.*, pages 1021–1026, 2005.

[129] W. S. Sarle. Stopped training and other remedies for overfitting. In *Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics*, pages 352–360, 1995.

[130] J. Schmidhuber. Evolutionary principles in self-referential learning. (on learning how to learn: The meta-meta-... hook.),diploma thesis. Master's thesis, Institut f. Informatik, Tech. Univ. Munich,, 1987.

[131] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[132] M. Schmidt and H. Lipson. Age-fitness pareto optimization. In R. Riolo, T. McConaghy, and E. Vladislavleva, editors, *Genetic Programming Theory and Practice VIII*, volume 8 of *Genetic and Evolutionary Computation*, pages 129–146. Springer New York, 2011.

[133] M. D. Schmidt and H. Lipson. Discovering a domain alphabet. In *GECCO*, pages 1083–1090, 2009.

[134] M. D. Schmidt and H. Lipson. Incorporating expert knowledge in evolutionary search: a study of seeding methods. In *GECCO*, pages 1091–1098, 2009.

[135] J. R. Sherrah, R. E. Bogner, and A. Bouzerdoum. The evolutionary pre-processor: Automatic feature extraction for supervised classification using genetic programming. In *In Proc. 2nd International Conference on Genetic Programming (GP-97*, pages 304–312. Morgan Kaufmann, 1997.

[136] S. Silva and E. Costa. Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. *Genetic Programming and Evolvable Machines*, 10:141–179, June 2009.

[137] M. Sips, B. Neubert, J. P. Lewis, and P. Hanrahan. Selecting good views of high-dimensional data using class consistency. *Computer Graphics Forum*, 28(3):831–838, 2009.

[138] M. Smith. *Using Genetic Programming for Feature Creation with a Genetic Algorithm Feature Selector*. PhD thesis, Bristol Institute of Technology University of the West of England, 2010.

[139] M. Smith and L. Bull. Improving the human readability of features constructed by genetic programming. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 1694–1701, New York, NY, USA, 2007. ACM.

[140] M. G. Smith and L. Bull. Genetic programming with a genetic algorithm for feature construction and selection. *Genetic Programming and Evolvable Machines*, 6(3):265–281, 2005.

[141] S. Smith. *A Learning System Based on Genetic Adaptive Algorithms*. PhD thesis, University of Pittsburgh, 1980.

[142] G. Smits and M. Kotanchek. Pareto-front exploitation in symbolic regression. pages 283–299, 13-15 May 2004.

[143] A. Tatu, G. Albuquerque, M. Eisemann, J. Schneidewind, H. Theisel, M. Magnor, and D. Keim. Combining automated analysis and visualization techniques for effective exploration of high-dimensional data. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (IEEE VAST)*, pages 59–66, Atlantic City, New Jersey, USA, 10 2009.

[144] A. Tatu, P. Bak, E. Bertini, D. Keim, and J. Schneidewind. Visual quality metrics and human perception: an initial study on 2d projections of large multidimensional data. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI '10, pages 49–56, New York, NY, USA, 2010. ACM.

[145] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[146] M. Tomassini, L. Vanneschi, F. Fernández, and G. G. Gil. A study of diversity in multipopulation genetic programming. In *Artificial Evolution*, pages 243–255, 2003.

[147] L. Trujillo, S. Silva, P. Legrand, and L. Vanneschi. An empirical study of functional complexity as an indicator of overfitting in genetic programming. In S. Silva, J. A. Foster, M. Nicolau, M. Giacobini, and P. Machado, editors, *Proceedings of the 14th European Conference on Genetic Programming, EuroGP 2011*, volume 6621 of *LNCS*, pages 263–274, Turin, Italy, 27-29 Apr. 2011. Springer Verlag.

[148] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, (1977).

[149] N. Q. Uy, N. X. Hoai, M. O'Neill, R. I. Mckay, and E. Galván-López. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12:91–119, June 2011.

[150] J. Valdes and A. Barton. Hybrid unsupervised/supervised virtual reality spaces for visualizing cancer databases: An evolutionary computation approach. In F. Sandoval, A. Prieto, J. Cabestany, and M. Graa, editors, *Computational and Ambient Intelligence*, volume 4507 of *Lecture Notes in Computer Science*, pages 1028–1035. Springer Berlin / Heidelberg, 2007.

[151] J. J. Valdes, R. Orchard, and A. J. Barton. Exploring medical data using visual spaces with genetic programming and implicit functional mappings. In *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*, GECCO '07, pages 2953–2960, New York, NY, USA, 2007. ACM.

[152] L. Vanneschi, M. Castelli, and S. Silva. Measuring bloat, overfitting and functional complexity in genetic programming. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 877–884, New York, NY, USA, 2010. ACM.

[153] L. Vanneschi and S. Gustafson. Using crossover based similarity measure to improve genetic programming generalization ability. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pages 1139–1146, New York, NY, USA, 2009. ACM.

[154] J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski. Information retrieval perspective to nonlinear dimensionality reduction for data visualization. *J. Mach. Learn. Res.*, 11:451–490, March 2010.

[155] A. Verma, X. Llorà, D. E. Goldberg, and R. H. Campbell. Scaling genetic algorithms using mapreduce. In *Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications*, ISDA '09, pages 13–18, Washington, DC, USA, 2009. IEEE Computer Society.

[156] M. Vlachos, C. Domeniconi, D. Gunopulos, G. Kollios, and N. Koudas. Non-linear dimensionality reduction techniques for classification and visualization. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 645–651, New York, NY, USA, 2002. ACM.

[157] E. Vladislavleva. *Model-based Problem Solving through Symbolic Regression via Pareto Genetic Programming.* PhD thesis, Tilburg University, Tilburg, the Netherlands, 2008.

[158] E. Vladislavleva, G. Smits, and D. den Hertog. Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. *IEEE Trans. Evolutionary Computation*, 13(2):333–349, 2009.

[159] C. Ware. *Information Visualization: Perception For Design*. Elsevier, 2004.

[160] A. R. Webb. *Statistical Pattern Recognition, 2nd Edition*. John Wiley & Sons, October 2002.

[161] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14:1–37, December 2007.

[162] D. Zaharie, D. Lungeanu, and F. Zamfirache. Interactive search of rules in medical data using multiobjective evolutionary algorithms. In *Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, GECCO '08, pages 2065–2072, New York, NY, USA, 2008. ACM.

[163] Y. Zhang and P. Rockett. A generic multi-dimensional feature extraction method using multiobjective genetic programming. *Evolutionary Computation*, 17(1):89–115, 2009.

[164] Y. Zhang and P. I. Rockett. A generic optimising feature extraction method using multiobjective genetic programming. *Applied Soft Computing*, 11(1):1087 – 1097, 2011.

[165] E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In K. Giannakoglou et al., editors, *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, pages 95–100. International Center for Numerical Methods in Engineering (CIMNE), 2002.

[166] J. Zupan, M. Novic, X. Li, and J. Gasteiger. Classification of multicomponent analytical data of olive oils using different neural networks. *Analytica Chimica Acta*, 292(3):219 – 234, 1994.