



HAL
open science

Prediction Performance and Problem Difficulty in Genetic Programming

Yuliana Martinez

► **To cite this version:**

Yuliana Martinez. Prediction Performance and Problem Difficulty in Genetic Programming. Artificial Intelligence [cs.AI]. ITT, Instituto tecnologico de Tijuana, 2016. English. NNT: . tel-01668769

HAL Id: tel-01668769

<https://inria.hal.science/tel-01668769>

Submitted on 20 Dec 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

SEP

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TIJUANA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA



PREDICCIÓN DE RENDIMIENTO Y DIFICULTAD DE
PROBLEMAS EN PROGRAMACIÓN GENÉTICA

TRABAJO DE TESIS PRESENTADO POR:
YULIANA SARAI MARTÍNEZ RAMOS

PARA OBTENER EL GRADO DE:
DOCTOR EN CIENCIAS DE LA INGENIERÍA

DIRECTOR DE TESIS:
DR. LEONARDO TRUJILLO REYES

TIJUANA, BAJA CALIFORNIA, MÉXICO.
JUNIO 2016

SEP

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TIJUANA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA



PREDICTION PERFORMANCE AND PROBLEM DIFFICULTY IN
GENETIC PROGRAMMING

THESIS SUBMITTED BY:

YULIANA SARAI MARTÍNEZ RAMOS

TO OBTAIN THE DEGREE OF:

DOCTOR IN ENGINEERING SCIENCES

ADVISOR:

DR. LEONARDO TRUJILLO REYES

TIJUANA, BAJA CALIFORNIA, MÉXICO.

JUNE 2016



"2015. Año del Generalísimo José María Morelos y Pavón"

Tijuana, B.C., 23 Mayo 2016

ASUNTO: Autorización de impresión de Trabajo de Tesis

Lic. Doroteo Luna Castañeda
Jefe del Dpto. de Servicios Escolares
Presente.


En lo referente al trabajo de tesis, "**Predicción de rendimiento y dificultad de problemas en programación genética**", presentado por la M.C. Yuliana Sarai Martínez Ramos, alumna del Doctorado en Ciencias de la Ingeniería, con número de Control **D02440897**, informamos a usted que después de una minuciosa revisión e intercambio de opiniones, los miembros del comité manifiestan APROBAR LA TESIS, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias por lo que se autoriza al interesado para que procesa de inmediato a la impresión del mismo.

ATENTAMENTE

"Por una Juventud Integrada al Desarrollo de México"


DR. PIERRICK LEGRAND
PRESIDENTE


DR. LEONARDO TRUJILLO REYES
SECRETARIO


DR. NOHÉ RAMÓN CÁZAREZ CASTRO
VOCAL


DR. MIGUEL ÁNGEL LÓPEZ RAMÍREZ
VOCAL


DR. VICTOR HUGO DÍAZ RAMÍREZ
VOCAL


DRA. YAZMÍN MALDONADO ROBLES
SUPLENTE

C.c.p. Oficina de Titulación
C.c.p. Director de Estudios de Posgrado e Investigación
C.c.p. Expediente
C.c.p. Intercedido
YMK



La estimación de la dificultad de problemas es un tema abierto en Programación Genética (GP). El objetivo de este trabajo es generar modelos que puedan predecir el desempeño esperado de un clasificador basado en GP cuando este es aplicado a tareas de prueba. Los problemas de clasificación son descritos usando características de un dominio específico, algunas de las cuales son propuestas en nuestro trabajo y estas características son dadas como entrada a los modelos predictivos. Nos referimos a estos modelos como predictores de desempeño esperado (PEPs, por sus siglas en inglés). Extendimos este enfoque usando un ensemble de predictores especializados (SPEPs, por sus siglas en inglés), dividiendo problemas de clasificación en grupos específicos y elegimos su correspondiente SPEP. Los predictores propuestos son entrenados usando problemas de clasificación sintéticos de 2D con conjunto de datos balanceados. Los modelos son entonces usados para predecir el desempeño de un clasificador de GP en problemas del mundo real antes no vistos los cuales son multidimensionales y desbalanceados. Además, este trabajo es el primero en proveer una predicción de rendimiento para un clasificador de GP sobre datos de prueba, mientras en trabajos previos se han enfocado en predecir el rendimiento para datos de entrenamiento. Por lo tanto, planteados como un problema de regresión simbólica son generados modelos predictivos exactos los cuales son resueltos con GP. Estos resultados son alcanzados usando características altamente descriptivas e incluyendo un paso de reducción de dimensiones el cual simplifica el proceso de aprendizaje y prueba. El enfoque propuesto podría ser extendido a otros algoritmos de clasificación y usarlo como base de un sistema experto de selección de algoritmos.

Palabras clave: Dificultad de problemas, Predicción de rendimiento esperado, Programación genética, Aprendizaje supervisado.

The estimation of problem difficulty is an open issue in Genetic Programming (GP). The goal of this work is to generate models that predict the expected performance of a GP-based classifier when it is applied to an unseen task. Classification problems are described using domain-specific features, some of which are proposed in this work, and these features are given as input to the predictive models. These models are referred to as predictors of expected performance (PEPs). We extend this approach by using an ensemble of specialized predictors (SPEP), dividing classification problems into groups and choosing the corresponding SPEP. The proposed predictors are trained using 2D synthetic classification problems with balanced datasets. The models are then used to predict the performance of the GP classifier on unseen real-world datasets that are multidimensional and imbalanced. This work is the first to provide a performance prediction of a GP system on test data, while previous works focused on predicting training performance. Accurate predictive models are generated by posing a symbolic regression task and solving it with GP. These results are achieved by using highly descriptive features and including a dimensionality reduction stage that simplifies the learning and testing process. The proposed approach could be extended to other classification algorithms and used as the basis of an expert system for algorithm selection.

Keywords: Problem difficulty, Prediction of expected performance, Genetic programming, Supervised learning.

To

my parents

ACKNOWLEDGEMENTS

Thanks God for allow me live beautiful moments and experiences with the people who love me.

I would like to thank all people who lived this dream with me.

To my dear parents, life is a constant struggle and becomes more comfortable hand of you, I have no words that represent how grateful I am for letting me dream, fly and reach my goals, I love you.

To my brothers, for always supporting me and loving me so much, I love you very much and I am very proud of you.

To my sister in a law, being like a sister, always listening to me, supporting me and giving me the most beautiful gift, my nephews.

To my nephews, they are the most beautiful thing happened to me, the best gift of a brother, thank you for so much love my beautiful boys, Julito and Cesarito.

To my dear grandparents, for their love, patience, so many beautiful moments I hope to have them with me much longer time, I love you.

To my family, aunts, uncles and cousins, thanks for their support all time.

To my friends, those who remain in good and bad times those that endure over time.

To my boyfriend, thanks for your love, support and understanding, just I love you.

I would like to thank my advisor, Dr. Leonardo Trujillo, for all his help and guidance that he has given me over the past six years. Thanks for his patience and for giving me always the best advices.

To Dr. Pierrick Legrand and family, for all your time, patience, advice and support.

To my dear TREE-LAB, the best people! Leonardo, Pierrick, Enrique, Emigdio, Luis, Victor, Perla, Uriel, Angel, Carlos and me.

To Dr. Francisco Fernández, Dr. Francisco Chávez, Dra. Sara Silva, being witnessed one of the most pleasant experiences I've had, to work with them in their institutions and know that different cultures.

To Dr. Edgar Galván-López, Dr. Victor Díaz-Ramírez, MC Arturo Sotelo, for their valuable comments on work and support.

I would like to express my gratitude to the members of my examination committee.

To the graduate program in Engineering Sciences, specially to the Dr. Luis Nestor Coria, thanks for the support.

To the Department of Electrical and Electronic Engineering at Instituto Tecnológico de Tijuana, for the great support given, MC Carlos Edgar Vázquez and Cinthya López.

I would like to thank to CONACYT for the opportunity to pursue my doctoral studies giving me a scholarship No. 226981.

Finally, I would like to thank to the FP7-Marie Curie-IRSES 2013 European Commission program through project ACoBSEC with Contract No. 612689, for the support provided.

Thank you

CONTENTS

| | |
|--|-----------|
| Resumen | I |
| Abstract | III |
| Acknowledgements | VII |
| Contents | IX |
| List of Figures | XI |
| List of Tables | XVII |
| 1. INTRODUCTION | 1 |
| 1.1. General Objective | 3 |
| 1.2. Particular Objectives | 3 |
| 1.3. Thesis Organization | 4 |
| 2. GENETIC PROGRAMMING | 5 |
| 2.1. Representation, Initialization and Operators | 5 |
| 2.2. Selection | 11 |
| 2.3. Terminals and Functions | 12 |
| 2.4. Fitness Function | 14 |
| 2.5. GP Parameters | 15 |
| 2.6. Training Set: <i>fitness-cases</i> | 16 |
| 2.6.1. Interleaved Sampling and related methods | 18 |
| 2.6.2. Lexicase Selection | 20 |
| 2.6.3. Keep-Worst Interleaved Sampling | 22 |
| 2.6.4. Comparison of Fitness-Case Sampling Methods | 23 |
| 3. CLASSIFICATION WITH GP | 49 |
| 3.1. Real-World Applications of GP Classifiers | 50 |
| 3.1.1. Static Range Selection GP Classifier (SRS-GPC) | 51 |
| 3.1.2. Probabilistic GP Classifier (PGPC) | 51 |
| 3.1.3. Experiments and Results | 52 |
| 4. THE ESTIMATION OF PROBLEM DIFFICULTY: RELATED WORK | 63 |
| 4.1. Evolvability Indicators | 63 |

| | |
|---|------------|
| 4.2. Performance Prediction | 65 |
| 5. PEP: PREDICTOR OF EXPECTED PERFORMANCE | 69 |
| 5.1. Synthetic Classification Problems | 70 |
| 5.2. PGPC Classification Error | 71 |
| 5.3. Preprocessing | 73 |
| 5.4. Feature Extraction | 75 |
| 5.5. Supervised Learning of PEP Models | 79 |
| 5.6. Testing the PEP models | 79 |
| 5.6.1. Testing on Synthetic Classification Problems . . . | 80 |
| 5.6.2. Testing on Real-World Classification Problems . . | 81 |
| 5.7. Comparative Study of EI and PEP | 87 |
| 6. SPEP: SPECIALIST PREDICTORS OF EXPECTED PERFORMANCE | 95 |
| 6.1. Grouping Problems based on PGPC Performance and Training SPEPs | 96 |
| 6.2. SPEP Selection | 97 |
| 6.3. Evaluation of SPEP Ensembles | 98 |
| 6.3.1. Ensemble-2 Solutions | 98 |
| 6.3.2. Ensemble-3 Solutions | 103 |
| 7. DISCUSSION | 111 |
| 8. CONCLUSIONS | 115 |
| Bibliography | 117 |
| A. NO FREE LUNCH THEOREMS (NFL) | 131 |
| A.1. No Free Lunch for Supervised Machine Learning | 131 |
| A.2. No Free Lunch for Search/Optimization | 132 |

LIST OF FIGURES

2.1. GP syntax tree representing $\max(x + x, x + 3 * y)$ 6

2.2. Multi-component program representation. 7

2.3. Creation of a full tree having maximum depth 2 using the full initialization method ($t = time$). 8

2.4. Creation of a five node tree using the grow initialization method with a maximum depth of 2 ($t = time$). A terminal is chosen at $t = 2$, causing the left branch of the root to be closed at that point even though the maximum depth had not been reached. 9

2.5. Example of subtree crossover. Note that the trees on the left are actually copies of the parents. So, their genetic material can freely be used without altering the original individuals. . . 10

2.6. Example of subtree mutation. 10

2.7. Box plot comparison about the test performance of the methods, from the best solution found for each benchmark symbolic regression problem over all thirty runs. 26

2.8. Box plot comparison about the overfitting performance of the methods, from the best solution found for each benchmark symbolic regression problem over all thirty runs. . . . 27

2.9. Box plot comparison about the average size performance of the methods, from the solutions found for each benchmark symbolic regression problem over all thirty runs. 28

2.10. Box plot comparison about the test performance of the methods, from the best solution found for each real-world regression problem over all thirty runs. 30

2.11. Box plot comparison about the overfitting performance of the methods, from the best solution found for each real-world regression problem over all thirty runs. 31

| | | |
|-------|---|----|
| 2.12. | Box plot comparison about the average size performance of the methods, from the solutions found for each real-world regression problem over all thirty runs. | 32 |
| 2.13. | Synthetic binary classification problems randomly generated using Gaussian mixture models with different amounts of class overlap, scattered over 2 dimensional space. | 36 |
| 2.14. | Box plot comparison about the test performance of the methods, from the best solution found for each synthetic classification problem over all thirty run. | 39 |
| 2.15. | Box plot comparison about the overfitting performance of the methods, from the best solution found for each synthetic classification problem over all thirty run. | 40 |
| 2.16. | Box plot comparison about the average size performance of the methods, from the solutions found for each synthetic classification problem over all thirty run. | 41 |
| 2.17. | Box plot comparison about the test performance of the methods, from the best solution found for each real-world classification problem over all thirty run. | 43 |
| 2.18. | Box plot comparison about the overfitting performance of the methods, from the best solution found for each real-world classification problem over all thirty run. | 44 |
| 2.19. | Box plot comparison about the average size performance of the methods, from the solutions found for each real-world classification problem over all thirty run. | 45 |
| 3.1. | Median classification error plotted against day used for training and amount of overlap for the SRS-GPC. Each row corresponds to each subject (top row is S_1 and bottom row S_4) and each column corresponds to a different segment duration; leftmost row is 1 sec., middle row 2 and rightmost 3. | 53 |
| 3.2. | Median classification error plotted against day used for training and amount of overlap for the PGPC. Each row corresponds to each subject (top row is S_1 and bottom row S_4) and each column corresponds to a different segment duration; leftmost row is 1 sec., middle row 2 and rightmost 3. . . | 56 |

| | | |
|------|---|----|
| 3.3. | Median sensitivity plotted against the day-used for training and the amount of overlap for the SRS-GPC. Each row corresponds to each subject (top row is S_1 and bottom row S_4) and each column corresponds to a different segment duration; leftmost row is 1 sec., middle row 2 and rightmost 3. . . | 58 |
| 3.4. | Median specificity plotted against the day used for training and the amount of overlap for the SRS-GPC. Each row corresponds to each subject (top row is S_1 and bottom row S_4) and each column corresponds to a different segment duration; leftmost row is 1 sec., middle row 2 and rightmost 3. . . | 59 |
| 3.5. | Median sensitivity plotted against the day used for training and the amount of overlap for the PGPC. Each row corresponds to each subject (top row is S_1 and bottom row S_4) and each column corresponds to a different segment duration; leftmost row is 1 sec., middle row 2 and rightmost 3. . . | 60 |
| 3.6. | Median specificity plotted against the day used for training and the amount of overlap for the PGPC. Each row corresponds to each subject (top row is S_1 and bottom row S_4) and each column corresponds to a different segment duration; leftmost row is 1 sec., middle row 2 and rightmost 3. . . | 61 |
| 5.1. | Block diagram of the proposed PEP approach. Given a classification problem, the goal is to predict the performance of a GP classifier on the test data, in this case PGPC. | 70 |
| 5.2. | The methodology used to build the PEP model. Given a set \mathcal{Q} of synthetic classification problems: (1) compute the CE_μ of PGPC on all problems; (2) apply a preprocessing for dimensionality reduction; (3) extract the feature vector β from the problem data; and (4) learn the predictive model using GP. | 70 |
| 5.3. | The scatter plots show examples of synthetic classification problems, specifying the CE_μ and standard deviation σ achieved by PGPC. These ordered from the lowest CE_μ (easiest depict in Fig. 5.3(a)) to the highest CE_μ (hardest depict in Fig. 5.3(f)). | 72 |

| | | |
|-------|---|----|
| 5.4. | Performance of PGPC over all 500 synthetic problems in \mathcal{Q} ; where: (a) shows the $CE\mu$ for each problem, ordered from the easiest to the hardest; and (b) shows the histogram over $CE\mu$ | 74 |
| 5.5. | Performance of PGPC over all 300 synthetic problems in $\mathcal{Q}' \subset \mathcal{Q}$; where: (a) shows the $CE\mu$ for each problem, ordered from the easiest to the hardest; and (b) shows the histogram over $CE\mu$ | 74 |
| 5.6. | These figures depict the complexity features used to describe each classification problem as suggested in Ho and Basu (2002), where: (a) Feature Efficiency (FE); (b) Class Distance Ratio (CDR); and (c) Volume of Overlap Region (VOR). | 77 |
| 5.7. | Scatter plots show the relationship between the $CE\mu$ (x-axis) and each descriptive feature (y-axis) for all problems $p \in \mathcal{Q}'$, where ρ specifies Pearson's correlation coefficient. | 78 |
| 5.8. | Figures show for synthetic problems, the performance prediction of the best PEP models evolved with the different feature set, each row belongs to each feature set: PEP-4F(top), PEP-5F(middle) and PEP-7F(bottom). | 82 |
| 5.9. | Histogram of imbalance percentage for the 40 real-world classification problems. | 85 |
| 5.10. | Performance of PGPC on the 40 real-world classification problems; where: (a) shows the $CE\mu$ for each problem; and (b) shows the histogram over $CE\mu$ | 85 |
| 5.11. | Scatter plots show for the real-world problems the relationship between the $CE\mu$ (x-axis) and each descriptive feature (y-axis). The legend specifies Pearson's correlation coefficient ρ | 86 |
| 5.12. | Figures show for real-world problems, the performance prediction of the best PEP models evolved with the different feature set, each row belongs to each feature set: PEP-4F(top), PEP-5F(middle) and PEP-7F(bottom). | 88 |

| | | |
|-------|--|-----|
| 5.13. | Scatter plot of the average classification error achieved by PGPC on each problem and the corresponding NSC value. Pearson's correlation coefficient $\rho = 0.02$ | 91 |
| 5.14. | Scatter plots that show the average performance of PGPC (x-axis) and the predicted performance of each PEP model (y-axis). The legend specifies Pearson's correlation coefficient $\rho = 0.86$ | 93 |
| 6.1. | Block diagram showing the proposed SPEP approach. The proposed approach is an extension of the basic PEP approach of Figure 5.1, with the additional ensemble approach, where problems are first classified into prespecified groups and based on this a corresponding specialized model (SPEP) is chosen to compute the PCE of PGPC on the test set. | 96 |
| 6.2. | The proposed groupings of classification problems used with the SPEP approach, showing the ranges of PGPC performance and the number of problems in each group. | 97 |
| 6.3. | Parallel coordinate plots dividing the set of problems into two groups, where each coordinate is given by a feature in β . Plots are shown for synthetic (a) and real-world problems (b). The plots on the left show a single line for each problem, while the plots on the right show the median values for each group. For clarity in the parallel plots, the features SD and CDR were rescaled to values between $[0, 1]$ | 99 |
| 6.4. | Performance prediction of the best Ensemble-2 solutions for each feature set: 4F(top), 5F(middle) and 7F (bottom). | 104 |
| 6.5. | Parallel coordinate plots dividing the set of problems into three groups, where each coordinate is given by a feature in β . Plots are shown for synthetic (a) and real-world problems (b). The plots on the left show a single line for each problem, while the plots on the right show the median values for each group. For clarity in the parallel plots, the features SD and CDR were rescaled to values between $[0, 1]$ | 105 |

| | | |
|------|---|-----|
| 6.6. | Performance prediction of the best Ensemble-3 solutions for each feature set: 4F(top), 5F(middle) and 7F (bottom). | 107 |
| 7.1. | Feature selection by the symbolic regression GP used to evolve all PEP and SPEP models, showing usage frequency over 100 runs: (a) bar plot of the total number of times that each feature appeared as a terminal element in the best models; and (b) median of the number of times that each feature appeared in each tree. | 112 |
| 7.2. | Scatter plots show the relationship between the percentage of the total variance explained by two principal components (x-axis) and the prediction error (y-axis), for all problems $p \in \mathcal{Q}'$, where the prediction error is the absolute difference between the $CE\mu$ and PCE, figure on the left show the PEP-4F model and figure on the right SPEP-2-5F, where ρ specifies Pearson's correlation coefficient. | 114 |

LIST OF TABLES

2.1. Examples of primitives in GP terminal set. 13

2.2. Examples of primitives in GP function set. 14

2.3. Five symbolic regression problems, originally published in Uy et al. (2011), and suggested as benchmark regression problems in McDermott et al. (2012) and Martínez et al. (2013). 25

2.4. Table with the parameters for GP used for the benchmark symbolic regression problems. 25

2.5. Table with the parameters for GP used for symbolic regression real-world problems. 29

2.6. Table show the median of 30 executions for testing, overfitting and size; bold indicates best. 33

2.7. Results of the Friedman test for the classification problems, showing the p-value after the Bonferroni-Holm correction for each pairwise comparison; bold indicates that the test rejects the null hypothesis at the $\alpha = 0.05$ significance level. 34

2.8. Real-world classification problems from Irvine (UCI) machine learning repository. 37

2.9. Table with the parameters for GP used for the classification problems. 37

2.10. Table show the median of 30 executions over testing, overfitting and size; bold indicates best. 38

2.11. Results of the Friedman test for the classification problems, showing the p-value after the Bonferroni-Holm correction for each pairwise comparison; bold indicates that the test rejects the null hypothesis at the $\alpha = 0.05$ significance level. 42

3.1. Parameters for the PGPC system used in the experimental tests. 54

5.1. Parameters for the PGPC algorithm. 73

- 5.2. Parameters for the GP used to derive PEP models for PGPC algorithm. 80
- 5.3. Three different features sets used as terminal elements for the symbolic regression GP algorithm. 80
- 5.4. Prediction performance of the evolved PEPs applied on the synthetic problems using each feature set (4F, 5F and 7F, see Table 5.3). Performance is given based on the RMSE and Pearson’s correlation coefficient, with bold indicating the best performance. 81
- 5.5. Real-world datasets from the UCI machine learning repository used in this work. 83
- 5.6. The 40 real-world binary classification problems based on the UCI datasets. 84
- 5.7. Prediction performance of the evolved PEPs applied on the real-world problems using each feature set (4F, 5F and 7F, see Table 5.3). Performance is given based on the RMSE and Pearson’s correlation coefficient, with bold indicating the best performance. 87
- 6.1. RMSE of the best evolved SPEP models, using different feature sets (first column). Performance is given based on training and testing set. Moreover, each SPEP-*i* corresponds to the *i* – *th* problem group but is tested on both problem groups, as specified in the fourth column. Bold indicates the best performance on each group. 100
- 6.2. Performance on the SPEP selection problem for all tested classifiers, showing the median classification error from 100 independent runs. The performance is given on the training and testing sets. Bold text indicates the best performance on each feature set. 101
- 6.3. Performance on the SPEP selection problem for all tested classifiers, showing the classification error of the best solution found, evaluated over all real-world problems, with bold indicating the best performance on each feature set. . . 101

| | | |
|------|---|-----|
| 6.4. | Ensemble-2 prediction accuracy using each feature set (4F, 5F and 7F), using the best evolved SPEPs and the best classifiers with each feature set. Performance is given based on the RMSE and Pearson’s correlation coefficient when evaluated on the synthetic and real-world problem sets; with bold indicating the best performance on real-world problems. | 102 |
| 6.5. | RMSE of the best evolved SPEP models, using different feature sets (first column). Performance is given based on training and testing set. Moreover, each SPEP- <i>i</i> corresponds to the <i>i</i> -th problem group but is tested on all problem groups, as specified in column 4. Bold text indicates best performance on each group. | 106 |
| 6.6. | Performance on the SPEP selection problem for all tested classifiers, showing the median classification error from 100 independent runs. The performance is given on the training and testing sets, with bold indicating the best performance on each feature set. | 108 |
| 6.7. | Performance on the SPEP selection problem for all tested classifiers, showing the classification error of the best solution found, evaluated over all real-world problems, with bold indicating the best performance on each feature set. . . | 108 |
| 6.8. | Ensemble-3 prediction accuracy using each feature set (4F, 5F and 7F), using the best evolved SPEPs and the best classifiers with each feature set. Performance is given based on the RMSE and Pearson’s correlation coefficient when evaluated on the synthetic and real-world problem sets; with bold indicating the best performance on real-world problems. | 109 |
| 7.1. | A comparison of each predictor approach; where bold indicates best performance. | 112 |

1

INTRODUCTION

Within the field of Evolutionary Computation (EC) (Eiben and Smith, 2003) it is not yet clear if a particular algorithm will perform well on a specific problem instance. The “No Free Lunch” (NFL) theorem (Wolpert and Macready, 1997) has provided valuable theoretical and conceptual insights, broadly stating that all search algorithms on average are equivalent when they are evaluated over all possible problems. On the other hand, the NFL theorem does not apply to many common domains of genetic programming (GP) (Poli et al., 2009), a promising theoretical insight that drives research to develop the best possible GP-based search. Nevertheless, it is by now evident that most GP-based systems tend to perform well on some problem instances while failing on others, with little understanding as to why or when either of those two scenarios will arise (Graff et al., 2013a; Graff and Poli, 2008; Martínez et al., 2012; Trujillo et al., 2011a,b).

The above issue can be described as the study of problem difficulty, which has been studied in different ways in EC and GP literature. Some methods focus on analyzing the properties of a problem’s fitness landscape (Kinnear, 1994). This can be done, for instance, by defining specific classes of functions (He et al., 2015), or by extracting high-level features (Graff et al., 2013a; Graff and Poli, 2008; Martínez et al., 2012; Trujillo et al., 2011a,b) or statistical properties (Clergue et al., 2002; Galván-López et al., 2008, 2010; Goldberg, 1987; Kimura, 1983; Rothlauf, 2006; Vanneschi et al., 2011; Verel et al., 2003) of the fitness landscape. In the case of standard tree-based GP, where search operators are applied in syntax space, the concept of a fitness landscape is difficult to define given that there is no clear way of de-

INTRODUCTION

termining a general concept of neighborhood for GP representations that are usually highly redundant, which limits the usefulness of such approaches. While some methods have been successfully applied to GP, these are mostly sampling-based techniques that attempt to infer specific types of structures within the underlying fitness landscape, such as: neutrality (Galván-López et al., 2008; Galván-López and Poli, 2006a,b; Poli and Galván-López, 2012; Yu and Miller, 2001), locality (Galván-López et al., 2010, 2011), ruggedness (Vanneschi et al., 2011), deception (Tomassini et al., 2005), fitness distance correlation (FDC) (Clergue et al., 2002; Tomassini et al., 2005), fitness clouds (Vanneschi et al., 2004) and negative slope coefficient (NSC) (Vanneschi et al., 2006, 2007). In this work, we refer to such methods as Evolvability Indicators (EIs), which are extensively reviewed in Malan and Engelbrecht (2013) and discussed in the following section.

One notable shortcoming of EIs is that they require an extensive sampling of the search space in order to compute them (Altenberg, 1997; Quick et al., 1998; Vanneschi et al., 2003, 2009). This is an important limitation: if we need to know when a particular problem is easy or difficult for an algorithm to solve it may just be easier to run the algorithm and observe its behavior and outcome. Therefore, some researchers have proposed predictive models that take the problem data (or a description of the data) as input, and produce as output a prediction of expected performance, we will refer to such methods as Predictors of Expected Performance (PEPs). Currently, the development of PEPs represents a minority of research devoted to problem difficulty in GP, with only a few recent works. In particular, Graff and Poli (Graff et al., 2013a; Graff and Poli, 2008, 2010, 2011; Graff et al., 2013b) have studied the development of such predictive models, for symbolic regression, Boolean and time-series problems. While their original work mostly focused on synthetic benchmarks (Graff and Poli, 2008), more recent contributions extended their approach to performance prediction in real-world problems (Graff et al., 2013a,b). However, in their approach it is necessary to have an extensive knowledge of the real-world problems in advance. Furthermore, their models are intended to predict the performance of the best solution found by GP on the train-

ing set of data, they did not address the prediction of performance on unseen test cases.

This work is an extension of our previous work (Trujillo et al., 2011a,b,c) where PEPs were first proposed for a GP classifier, making several methodological and experimental contributions. First, the PEP models are produced using only simple 2D synthetic datasets that are randomly generated. Second, the PEP models are used to predict the performance of the GP classifier on the test set of data, while previous works mostly focused on predicting performance on the training or learning set (Graff et al., 2013a; Graff and Poli, 2008, 2010, 2011; Graff et al., 2013b). Third, accurate predictions are obtained on unseen real-world problems that are multidimensional and contain imbalanced data. On the other hand, previous works (Graff et al., 2013a; Graff and Poli, 2008, 2010, 2011; Graff et al., 2013b; Trujillo et al., 2011a,b,c) used the same type of problems (either synthetic or real) for both training and testing. Fourth, to increase PEP accuracy this work presents an ensemble approach using specialized PEP models called SPEPs. Each SPEP is trained to predict performance within a specific range of classification error. To do so, we use a two-tier approach, where each problem is first classified into a specific group, and then prediction is obtained from the corresponding SPEP which was trained for that group of problems. Finally, it is reasonable to state that the proposed approach could be applied to predict the performance of GP on other learning problems.

1.1 GENERAL OBJECTIVE

Build models to predict the expected performance and categorize its difficulty into classification problems using Genetic Programming.

1.2 PARTICULAR OBJECTIVES

1. Characterize the classification problems using descriptive measures.

INTRODUCTION

2. Build models to predict the expected performance using synthetic classification problems.
3. Categorize the classification problems (in *easy* or *hard*).
4. Evaluate the methodology on real-world classification problems, a more interesting and realistic scenario.

1.3 THESIS ORGANIZATION

The remainder of this thesis proceeds as follows.

Chapter 2 shows an overview about Genetic Programming.

Chapter 3 provides a short survey of GP-based classification and presents two implementations of GP classifiers on real-world scenarios.

Chapter 4 reviews the state of art both Evolvability Indicators and Predictors of Expected Performance.

The basic PEP strategy is outlined and evaluated in Chapter 5.

Afterwards, Chapter 6 introduces the proposed ensemble strategy based on SPEPs and provides experimental results.

Chapter 7 presents a discussion related to results and contributions.

Finally, Chapter 8 contains conclusions and future work.

2

GENETIC PROGRAMMING

The goal of having computers automatically solve problems is central to artificial intelligence, machine learning and the broad area encompassed by what Turing called “machine intelligence” (Turing, 1950). Machine learning pioneer Arthur Samuel, in his 1983 talk entitled “AI: Where It Has Been and Where It Is Going” (Samuel, 1983), stated that the main goal of the fields of machine learning and artificial intelligence is:

“to get machines to exhibit behaviour, which if done by humans, would be assumed to involve the use of intelligence.”

Genetic Programming (GP) is an extension of Genetic Algorithms (GA's) and owes its origin to the work of John Koza (Koza, 1992). It basically works as GA's, with the major difference that individuals to be evolved are not fixed length strings of characters, but, generally speaking, *computer programs*. GP is an evolutionary computation (EC) technique that automatically solves problems without requiring the user to know or specify the form or structure of the solution in advance (Poli and McPhee, 2008). At the most abstract level GP is a systematic, domain-independent method for getting computers to solve problems automatically starting from a high-level statement of what needs to be done.

2.1 REPRESENTATION, INITIALIZATION AND OPERATORS

In GP, programs are usually expressed as syntax trees rather than as lines of code. For example Figure 2.1 shows the tree representation

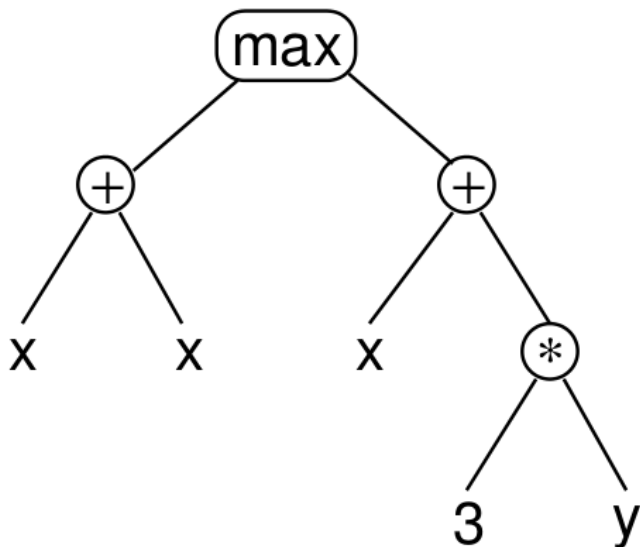


Figure 2.1: GP syntax tree representing $\max(x + x, x + 3 * y)$.

of the program $\max(x + x, x + 3 * y)$. The variables and constants in the program (x, y and 3) are leaves of the tree. In GP they are called terminals, whilst the arithmetic operations ($+, *$ and \max) are internal nodes called functions. The sets of allowed functions and terminals together form the primitive set of a GP system. In more advanced forms of GP, programs can be composed of multiple components (e.g., sub-routines). In this case the representation used in GP is a set of trees (one for each component) grouped together under a special root node that acts as glue, as illustrated in Figure 2.2 we will call these (sub)trees branches. The number and type of the branches in a program, together with certain other features of their structure, form the architecture of the program.

Like in other evolutionary algorithms, in GP the individuals in the initial population are typically randomly generated. There are a number of different approaches to generating this random initial popula-

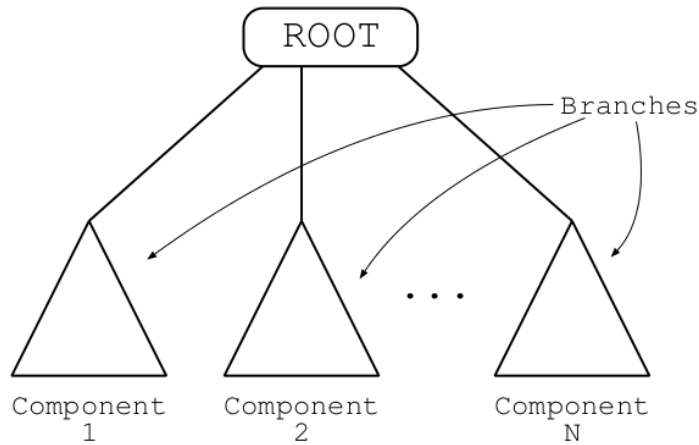


Figure 2.2: Multi-component program representation.

tion. Here we will describe two of the simplest (and earliest) methods (the *full* and *grow* methods), and a widely used combination of the two known as *Ramped half-and-half*.

In both the *full* and *grow* methods, the initial individuals are generated so that they do not exceed a user specified maximum depth. The depth of a node is the number of edges that need to be traversed to reach the node starting from the tree's root node (which is assumed to be at depth 0). The depth of a tree is the depth of its deepest leaf (e.g., the tree in Figure 2.1 has a depth of 3). In the full method (so named because it generates full trees, i.e. all leaves are at the same depth) nodes are taken at random from the function set until the maximum tree depth is reached. Figure 2.3 shows a series of snapshots of the construction of a full tree of depth 2. The children of the $*$ and $/$ nodes must be leaves or otherwise the tree would be too deep. Thus, at both steps $t = 3$, $t = 4$, $t = 6$ and $t = 7$ a terminal must be chosen (x , y , 1 and 0, respectively). Although, the full method generates trees where all the leaves are at the same depth, this does not necessarily mean that all initial trees will have an identical number of nodes (often referred to as the size of a tree) or the same shape. This only happens, in fact, when

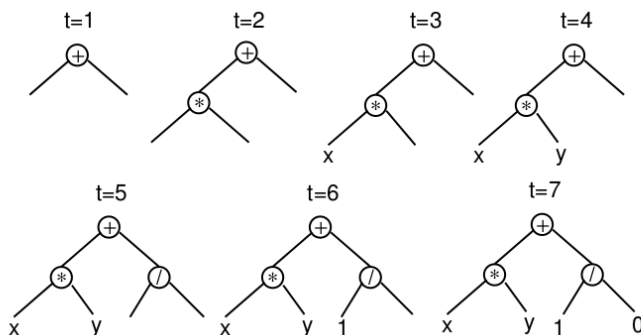


Figure 2.3: Creation of a full tree having maximum depth 2 using the full initialization method ($t = time$).

all the functions in the primitive set have an equal arity. Nonetheless, even when mixed-arity primitive sets are used, the range of program sizes and shapes produced by the full method may be rather limited. The grow method, on the contrary, allows for the creation of trees of more varied sizes and shapes. Nodes are selected from the whole primitive set (i.e., functions and terminals) until the depth limit is reached. Once the depth limit is reached only terminals may be chosen (just as in the full method). Figure 2.4 illustrates this process for the construction of a tree with depth limit 2. Here the first argument of the $+$ root node happens to be a terminal. This closes off that branch preventing it from growing any more before it reached the depth limit. The other argument is a function ($-$), but its arguments are forced to be terminals to ensure that the resulting tree does not exceed the depth limit.

GP departs significantly from other evolutionary algorithms in the implementation of the operators of crossover and mutation. The most commonly used form of crossover is subtree crossover. Given two parents, subtree crossover randomly (and independently) selects a crossover point (a node) in each parent tree. Then, it creates the offspring by replacing the subtree rooted at the crossover point in a copy of the first parent with a copy of the subtree rooted at the crossover

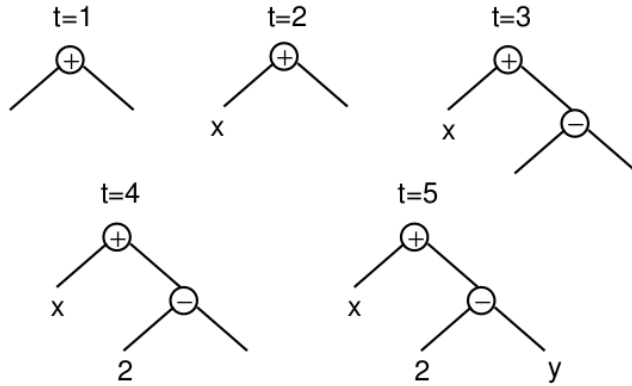


Figure 2.4: Creation of a five node tree using the grow initialization method with a maximum depth of 2 ($t = \text{time}$). A terminal is chosen at $t = 2$, causing the left branch of the root to be closed at that point even though the maximum depth had not been reached.

point in the second parent, as illustrated in Figure 2.5 Copies are used to avoid disrupting the original individuals. This way, if selected multiple times, they can take part in the creation of multiple offspring programs. Note that it is also possible to define a version of crossover that returns two offspring, but this is not commonly used. Often crossover points are not selected with uniform probability. Typical GP primitive sets lead to trees with an average branching factor (the number of children of each node) of at least two, so the majority of the nodes will be leaves. Consequently the uniform selection of crossover points leads to crossover operations frequently exchanging only very small amounts of genetic material (i.e., small subtrees); many crossovers may in fact reduce to simply swapping two leaves. To counter this, Koza (1992) suggested the widely used approach of choosing functions 90% of the time and leaves 10% of the time. Many other types of crossover and mutation of GP trees are possible.

The most commonly used form of mutation in GP (which we will call subtree mutation) randomly selects a mutation point in a tree and

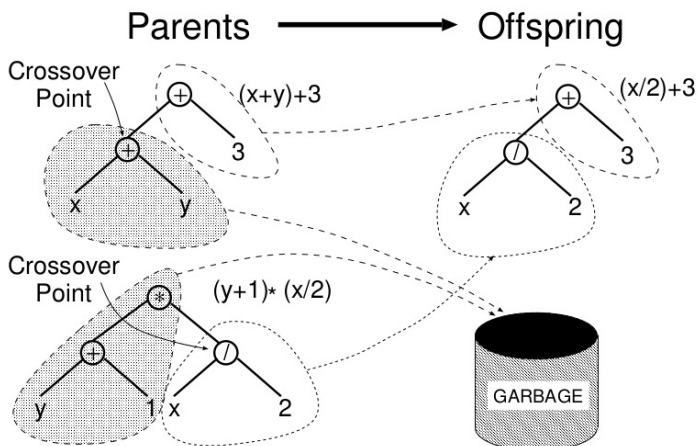


Figure 2.5: Example of subtree crossover. Note that the trees on the left are actually copies of the parents. So, their genetic material can freely be used without altering the original individuals.

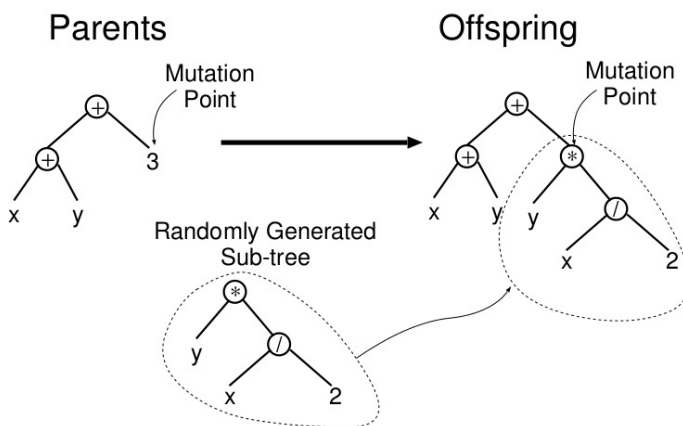


Figure 2.6: Example of subtree mutation.

substitutes the subtree rooted there with a randomly generated subtree. This is illustrated in Figure 2.6. Subtree mutation is sometimes imple-

mented as crossover between a program and a newly generated random program; this operation is also known as “headless chicken” crossover (Angeline, 1997). Another common form of mutation is point mutation, which is GP’s rough equivalent of the bit-flip mutation used in genetic algorithms (Goldberg, 1989). In point mutation, a random node is selected and the primitive stored there is replaced with a different random primitive of the same arity taken from the primitive set. If no other primitives with that arity exist, nothing happens to that node (but other nodes may still be mutated). When subtree mutation is applied, this involves the modification of exactly one subtree. Point mutation, on the other hand, is typically applied on a per-node basis. That is, each node is considered in turn and, with a certain probability, it is altered as explained above. This allows multiple nodes to be mutated independently in one application of point mutation. The choice of which of the operators described above should be used to create an offspring is probabilistic. Operators in GP are normally mutually exclusive (unlike other evolutionary algorithms where offspring are sometimes obtained via a composition of operators). Their probability of application are called operator rates. Typically, crossover is applied with the highest probability, the crossover rate often being 90% or higher. On the contrary, the mutation rate is much smaller, typically being in the region of 1%. When the rates of crossover and mutation add up to a value of probability (τ) which is less than 100%, an operator called reproduction is also used, with a rate of $1 - \tau$. Reproduction simply involves the selection of an individual based on fitness and the insertion of a copy of it in the next generation.

2.2 SELECTION

As with most evolutionary algorithms, genetic operators in GP are applied to individuals that are probabilistically selected based on fitness. That is, better individuals are more likely to have more child programs than inferior individuals. The most commonly employed method for selecting individuals in GP is tournament selection, which

is discussed below, followed by fitness-proportionate selection, but any standard evolutionary algorithm selection mechanism can be used. In tournament selection a number of individuals are chosen at random from the population. These are compared with each other and the best of them is chosen to be the parent. When doing crossover, two parents are needed and, so, two selection tournaments are made. Note that tournament selection only looks at which program is better than another. It does not need to know how much better. This effectively automatically rescales fitness, so that the selection pressure on the population remains constant. Thus, a single extraordinarily good program cannot immediately swamp the next generation with its children; if it did, this would lead to a rapid loss of diversity with potentially disastrous consequences for a run. Conversely, tournament selection amplifies small differences in fitness to prefer the better program even if it is only marginally superior to the other individuals in a tournament. An element of noise is inherent in tournament selection due to the random selection of candidates for tournaments. So, while preferring the best, tournament selection does ensure that even average-quality programs have some chance of having children. Since tournament selection is easy to implement and provides automatic fitness rescaling, it is commonly used in GP. Considering that selection has been described many times in the evolutionary algorithms literature, we will not provide details of the numerous other mechanisms that have been proposed. Goldberg (1989), for example, describes fitness-proportionate selection, stochastic universal sampling and several others.

2.3 TERMINALS AND FUNCTIONS

While it is common to describe GP as evolving programs, GP is not typically used to evolve programs in the familiar Turing-complete languages humans normally use for software development. It is instead more common to evolve programs (or expressions or formulae) in a more constrained and often domain-specific language. The first two preparatory steps, the definition of the terminal and function sets, spec-

Table 2.1: Examples of primitives in GP terminal set.

| Terminal Set | |
|--------------------------|---------------|
| Kind of Primitive | Example (s) |
| <i>Variables</i> | x, y |
| <i>Constant values</i> | 3, 0.45 |
| <i>0-arity functions</i> | rand, go_left |
| .. | ... |

ify such a language. That is, together they define the ingredients that are available to GP to create computer programs.

The terminal set may consist of:

- *the program's external inputs.* These typically take the form of named variables (*e.g.*, x, y).
- *functions with no arguments.* These may be included because they return different values each time they are used, such as the function *rand()* which returns random numbers, or a function *dist to wall()* that returns the distance to an obstacle from a robot that GP is controlling. Another possible reason is because the function produces side effects. Functions with side effects do more than just return a value: they may change some global data structures, print or draw something on the screen, control the motors of a robot, etc.
- *constants.* These can be pre-specified, randomly generated as part of the tree creation process, or created by mutation.

Using a primitive such as *rand* can cause the behaviour of an individual program to vary every time it is called, even if it is given the same inputs. This is desirable in some applications. However, we more often want a set of fixed random constants that are generated as part of the process of initializing the population. This is typically accomplished by introducing a terminal that represents an ephemeral random constant. Every time this terminal is chosen in the construction of an initial tree (or a new subtree to use in an operation like mutation), a

Table 2.2: Examples of primitives in GP function set.

| Function Set | |
|--------------------------|--------------------|
| Kind of Primitive | Example (s) |
| <i>Arithmetic</i> | +, *, / |
| <i>Mathematical</i> | sin, cos, exp |
| <i>Boolean</i> | AND, OR, NOT |
| <i>Conditional</i> | IF-THEN-ELSE |
| <i>Looping</i> | FOR, REPEAT |
| .. | ... |

different random value is generated which is then used for that particular terminal, and which will remain fixed for the rest of the run. The function set used in GP is typically driven by the nature of the problem domain. In a simple numeric problem, for example, the function set may consist of merely the arithmetic functions (+, -, *, /). However, all sorts of other functions and constructs typically encountered in computer programs can be used. Tables 2.1 and 2.2 shows a sample of some of the functions one sees in the GP literature. Sometimes the primitive set includes specialized functions and terminals which are designed to solve problems in a specific problem domain. For example, if the goal is to program a robot to mop the floor, then the function set might include such actions as move, turn, and swish-the-mop.

2.4 FITNESS FUNCTION

The first two preparatory steps define the primitive set for GP, and therefore indirectly define the search space GP will explore. This includes all the programs that can be constructed by composing the primitives in all possible ways. However, at this stage, we still do not know which elements or regions of this search space are good. I.e., which regions of the search space include programs that solve, or approximately solve, the problem. This is the task of the fitness measure, which is our primary (and often sole) mechanism for giving a high-level statement

of the problem's requirements to the GP system. For example, suppose the goal is to get GP to synthesize an amplifier automatically. Then the fitness function is the mechanism which tells GP to synthesize a circuit that amplifies an incoming signal. (As opposed to evolving a circuit that suppresses the low frequencies of an incoming signal, or computes its square root, etc. etc.) Fitness can be measured in many ways. For example, in terms of: the amount of error between its output and the desired output; the amount of time (fuel, money, etc.) required to bring a system to a desired target state; the accuracy of the program in recognising patterns or classifying objects; the payoff that a game-playing program produces; the compliance of a structure with user-specified design criteria.

2.5 GP PARAMETERS

The most important control parameter is the population size. Other control parameters include the probabilities of performing the genetic operations, the maximum size for programs and other details of the run. It is impossible to make general recommendations for setting optimal parameter values, as these depend too much on the details of the application. However, genetic programming is in practice robust, and it is likely that many different parameter values will work. As a consequence, one need not typically spend a long time tuning GP for it to work adequately. It is common to create the initial population randomly using *ramped half-and-half* with a depth range of 2–6. The initial tree sizes will depend upon the number of the functions, the number of terminals and the arities of the functions. However, evolution will quickly move the population away from its initial distribution. Traditionally, 90% of children are created by subtree crossover. However, the use of a 50–50 mixture of crossover and a variety of mutations also appears to work well.

2.6 TRAINING SET: *fitness-cases*

GP is widely used to generate mathematical functions, or operators, that solve symbolic regression and classification problems, which can be stated as follows. The goal is to search for the symbolic expression $K : \mathbb{R}^p \rightarrow \mathbb{R}$ that best fits a particular training set $\mathbb{T} = \{I_1, I_2, \dots, I_n\}$, where $I_i = (v_i, \nu_i)$ of n input-output pairs with $v_i \in \mathbb{R}^p$ and $\nu_i \in \mathbb{R}$, stated as

$$K \leftarrow \underset{K \in \mathbb{G}}{\text{arg min}} f(K(v_i), \nu_i) \text{ with } i = 1, \dots, n, \quad (2.1)$$

where \mathbb{G} is the solution or syntactic space defined by the primitive set \mathbb{P} of functions and terminals, f is the fitness function which is based on the difference between a program's output $K(v_i)$ and the desired output ν_i . Each input-output pair (v_i, ν_i) is referred to as a fitness-case. As stated above, GP traditionally uses the entire set of fitness-cases \mathbb{T} to determine a program's fitness. Recent works, however, have focused on evolving solutions by only using a subset of fitness-cases $\mathbb{S}_{\mathcal{G}} \subseteq \mathbb{T}$ to determine the fitness of each candidate solution K at a given generation \mathcal{G} ; in some cases reducing the number of considered fitness-cases to a single one. This, of course, can produce one obvious benefit, a reduction in computational cost during fitness evaluation, which is usually the main computational bottleneck in a GP system. However, the motivation for most approaches has been varied. For instance, some methods focus on reducing computational costs during fitness evaluation (Gathercole and Ross, 1994a,b) or reducing overfitting and improving generalization (Gonçalves and Silva, 2013), others have addressed the issue of problem modality (Spector, 2012) or attempted to promote novel solutions that can solve every fitness case (Martínez et al., 2013), particularly the most difficult ones (Gathercole and Ross, 1994a,b; Martínez et al., 2014).

GP traditionally uses the Entire Training Set (ETS) of fitness-cases \mathbb{T} to determine a program's fitness. Recent works, however, have focused on evolving solutions by only using a subset of fitness-cases $\mathbb{S}_{\mathcal{G}} \subseteq \mathbb{T}$ to determine the fitness of each candidate solution K at a given generation \mathcal{G} ; in some cases reducing the number of considered fitness-

cases to a single one. This, of course, can produce one obvious benefit, a reduction in computational cost during fitness evaluation, which is usually the main computational bottleneck in a GP system. However, the motivation for most approaches has been varied. For instance, some methods focus on reducing computational costs during fitness evaluation (Gathercole and Ross, 1994a,b) or reducing overfitting and improving generalization (Gonçalves and Silva, 2013), others have addressed the issue of problem modality (Spector, 2012) or attempted to promote novel solutions that can solve every fitness case (Martínez et al., 2013), particularly the most difficult ones (Gathercole and Ross, 1994a,b; Martínez et al., 2014). In what follows, we review these methods and describe their main details.

Dynamic Training Subset Selection (DTSS) was proposed in Gathercole and Ross (1994a,b), and probably should be considered as the first fitness-case sampling method proposed in GP literature. In their original work, Gathercole and Ross (1994a,b) developed their method for classification problems, outperforming the basic GP approach, but it was only tested on a single problem instance. In each generation \mathcal{G} DTSS performs two passes through the entire training set of fitness-cases. It assigns a weight w_i to each fitness-case I_i computed by

$$w_i(\mathcal{G}) = D_i(\mathcal{G})^d + A_i(\mathcal{G})^a \quad (2.2)$$

where D is a function that measures the difficulty of I_i , A is an age factor that measures the number of generations \mathcal{G} since I_i was last selected (D and A are reset to zero once I_i is selected), while a and d are parameters that need to be set by the user. Afterwards a total of \mathcal{F} fitness are selected using a probability that is proportional to its associated weight, given by $\Phi = \frac{w_i}{\sum_{j=1}^n w_j}$ with n the total number of fitness-cases. In their original work, D_i represented the total number of times that a particular fitness case was misclassified by the individuals in the population. While the authors reported strong results on some tests, DTSS has not been extensively benchmarked on a variety of problems. Moreover, there are several shortcomings with DTSS. In particular, it requires several parameters to be tuned; the difficulty score is not triv-

Algorithm 1: Interleaved Sampling (IS)

Deterministically interleaving between using a single or all training fitness cases at each generation:

- (1) Initialize:
 - (a) Entire training set $\mathbb{T} = \{I_1, I_2 \dots I_n\}$.
 - (2) First generation:
 - (a) Evaluate population using \mathbb{T} .
 - (3) Loop on the remaining generations:
 - (a) Odd generations:
 - Generate an integer random number r , where $r \in [1, n]$.
 - Evaluate population using I_r .
 - (b) Even generations:
 - Evaluate population using \mathbb{T} .
-

ially extrapolated to other problems, particularly symbolic regression; and, it is more computationally costly than standard GP since at every generation the algorithm requires two passes over the entire training set of fitness-cases.

2.6.1 *Interleaved Sampling and related methods*

Interleaved Sampling (IS) (Gonçalves and Silva, 2011) is a deterministic-based sampling method, which uses the entire training set to compute fitness in some generations and uses a single fitness-case in others. This approach was motivated by the idea of balancing learning and overfitting through the interleaving of fitness-cases, attempting to elude local optima. Determining in which generation to

Algorithm 2: Random Interleaved Sampling (RIS)

Probabilistically interleaving between using a single or all training fitness cases at each generation:

- (1) Initialize:
 - (a) Entire training set $\mathbb{T} = \{I_1, I_2 \dots I_n\}$.
 - (b) Interleave probability $\delta \in [0, 1]$.
 - (2) Loop for every generation:
 - (a) Generate a real random number $rn \in [0, 1]$.
 - (b) If $rn \leq \delta$ then
 - Generate an integer random number r , where $r \in [1, n]$.
 - Evaluate population using I_r .
 - (c) Otherwise
 - Evaluate population using \mathbb{T} .
-

use a single fitness-case, and in which one to use all of them, is an integral part of the algorithm design. In Gonçalves and Silva (2013), the authors present two variants that achieved the best results, IS and Random IS or RIS with the probability value of 75%. IS uses the entire training set in odd numbered generations, and uses a single randomly chosen fitness-case on even numbered generations, see the Algorithm 1. RIS, on the other hand, uses a probabilistic decision at the beginning of each generation to determine if all of the fitness-cases are used or a single randomly chosen fitness-case, see the Algorithm 2. In Gonçalves and Silva (2013), RIS exhibited the best performance with the probability of using a single fitness case set to $\delta = 0.75$. These methods are related to the approach presented in Lasarczyk et al. (2004), that also selects a different subset of fitness-cases at each generation.

Algorithm 3: Lexicase Selection (LEX)

For parent selection:

- (1) Initialize:
 - (a) Set **candidates** to be the entire population.
 - (b) Set **cases** to be a list of all of the fitness cases in random order.
 - (2) Loop:
 - (a) Set **candidates** to be the subset of the current candidates that have exactly the best fitness of any individual currently in **candidates** for the first case in **cases**.
 - (b) If **candidates** or **cases** contains just a single element then return the first individual in **candidates**.
 - (c) Otherwise remove the first case from cases and go to Loop.
-

2.6.2 Lexicase Selection

Spector (2012) proposed the concept of modality to describe problems for which an optimal solution must exhibit different modes of operations; i.e., solutions must exhibit distinct behaviors based on contextual information that is provided implicitly by the input data (Trujillo et al., 2013). Spector (2012) points out that, in general, GP systems are generally limited to problems for which solution programs perform similar actions for all possible inputs, but real-world problems will normally require more complex solutions, that change their behavior depending on context.

To solve such problems, Spector (2012) presents the Lexicase Selection (LEX) method for parent selection, which allows each fitness case to possibly be the main source of selective pressure at any given parent selection event. During evolution, LEX selects parents by starting

Algorithm 4: Keep-Worst Interleaved Sampling (KW-IS)

Deterministically interleaving between using a subset of the most difficult or all fitness cases at each generation:

- (1) Initialize:
 - (a) Entire training set $\mathbb{T} = \{I_1, I_2 \dots I_n\}$.
 - (b) Percentil value error $\rho \in [1, 100]$.
 - (2) First generation:
 - (a) Evaluate population using \mathbb{T} .
 - (b) Construct the subset φ of the *most difficult fitness cases* using the ρ value.
 - (3) Loop on the remaining generations:
 - (a) Odd generations:
 - Evaluate population using φ .
 - (b) Even generations:
 - (a) Evaluate population using \mathbb{T} .
 - (b) Construct the subset φ of the *most difficult fitness cases* using the ρ value.
-

with a pool of candidate parents, and removing candidates based on the performance achieved on a single fitness-case. LEX is elitist, all of the individuals that do not achieve the best performance are removed. This process is repeated using another fitness-case, until only one individual remains, which is then returned as the selected parent. In the basic implementation, the initial pool of candidates is composed by the entire population and the fitness-cases are ordered randomly each time a parent is selected. LEX resembles a lexicographic ordering of a character string, where the first fitness-case has the largest effect in choosing the parent, then the next fitness-case acts as tie-breaker, and

so on. This means that each fitness-case has a chance to be deciding factor in determining which individuals are used to produce offspring at any given parent selection event, see Algorithm 3.

2.6.3 *Keep-Worst Interleaved Sampling*

Based on Martínez et al. (2013) and Gonçalves and Silva (2013) we proposed a new fitness-case sampling method called Keep-Worst Interleaved Sampling (KW-IS) (Martínez et al., 2014). This proposed method is based on the general methodology of the Novelty Search-based ϵ – *descriptor* proposed in Martínez et al. (2013) but it is also common in other learning paradigms such as AdaBoost, for example, where solution design is adjusted based on the most difficult training data samples. KW-IS is similar to IS, using the entire set of fitness-cases in some generations, just like IS would. However, in the remaining generations, fitness-cases are not chosen randomly. Instead, the goal is to bias selective pressure towards individuals that exhibit good performance on the most difficult fitness-cases. Therefore, the fitness-cases are ordered based on difficulty. Afterwards, the ρ value show the percentage of most difficult fitness-cases are used to determine fitness in the next generation, see Algorithm 4. The best performance of this parameter was achieved with $\rho = 90$. For symbolic regression problems the difficulty of a single fitness case is given by the average absolute error of the entire population in a given generation. Where we take the fitness-cases that have a larger error than the error found in the percentile given the value of ρ . On the other hand, for classification the difficulty of a fitness-case is computed as done in DTSS, by the total number of individuals in the population that misclassified it. Therefore, in the same way we will take the fitness-cases that have been misclassified by ρ percent of the population.

KW-IS is closely related to DTSS, since it also focuses on reducing the size of the training set used in each generation by concentrating on a small subset of the most difficult fitness-cases. However, KW-IS requires less parameters to tune, it only performs a single pass of the

training set \mathbb{T} at each generation \mathcal{G} by relying on the estimated difficulty given in generation $\mathcal{G} - 1$, and it can be used directly on symbolic regression problems.

2.6.4 *Comparison of Fitness-Case Sampling Methods*

In order to provide a comprehensive evaluation of the fitness case sampling methods described above we perform experiments using a tree-based GP algorithm with standard subtree crossover and subtree mutation, as originally proposed by Koza (2010). With this GP system, we test IS, RIS, LEX and KW-IS on two of the most common problems on which GP is used, symbolic regression and classification, using benchmark and real-world problems. Moreover, a standard GP search (GP-STD) is included as the control method.

None of the algorithms have been extensively studied or compared besides our previous study reported in Martínez et al. (2014), however that work only performed an informal comparison and only considered symbolic regression problems. The algorithms are compared based on the following criteria. First, performance of the best solution found during training evaluated on the test set, a standard evaluation measure. Second is overfitting, here measured by the difference between the training error of the best solutions and its respective test error. Finally, one of the most important open issues with GP is the bloat phenomenon, where the average size of the population increases disproportionately relative to the improvements achieved in terms of fitness (Silva and Costa, 2009). Therefore, we also compare the methods based on the average size of the population in the final generation, given by the number of nodes of each tree.

In all problems a total of thirty independent runs are performed, with different, and randomly chosen, training and testing sets. Results are analyzed using rank statistics and nonparametric tests, since machine learning algorithms do not tend to produce normally distributed samples (Trawinski et al., 2012; Derrac et al., 2011). Therefore, the Friedman test multiple comparison (Friedman, 1937) is used to com-

pare all of the algorithms on each problem, with the null hypothesis that the mean rank of all methods is the same; the p-value of each test is reported and the significance level is set at $\alpha = 0.05$. Afterwards, a post-hoc procedure is used, performing pairwise comparisons between all methods using the Friedman test with the Bonferroni-Holm correction (Holm, 1979) of the p-value (considering six methods), under the null hypothesis that each pair of samples share equal medians. The results of all tests are presented in tables with the corresponding p-values. Moreover, a summary of the medians of each measure (test fitness, average size and overfitting) are presented, using bold to indicate that a given method achieved the best performance on a given problem. For instance, if the performance of all methods are presented in bold, this means that no statistical difference was detected. Similarly, if one (or several) result(s) is (are) in bold, this means that the method(s) achieved significantly different performance compared to the other (non-bold) methods. Box plots are used to graphically illustrate the behavior of each method. Finally, all algorithms were implemented using the GPLab Matlab toolbox (Silva and Almeida, 2003) and all statistical test were performed using the Matlab statistical toolbox.

2.6.4.1 *Symbolic Regression Problems*

Five benchmark problems are used for symbolic regression, originally published in Uy et al. (2011), and suggested in McDermott et al. (2012) and Martínez et al. (2013); these problems are summarized in Table 2.3. The experimental parameters used with these problems are given in Table 2.4, and fitness is calculated as the root mean square error between predicted and expected outputs specified in the training set.

Moreover, to get a better assessment of each method in more difficult scenarios, six problems are used to evaluate the algorithms; these are: Toxicity, Plasma Protein Binding, Bioavailability, Concrete, Housing and Yacht. The first three problems are described in Gonçalves and Silva (2013) and the remaining three from the University of California, Irvine (UCI) machine learning repository (Lichman, 2013), which

Table 2.3: Five symbolic regression problems, originally published in Uy et al. (2011), and suggested as benchmark regression problems in McDermott et al. (2012) and Martínez et al. (2013).

| Problem | Function | Fitness/Test cases |
|------------------|-----------------------------|--|
| f_1 -Benchmark | $x^4 + x^3 + x^2 + x$ | 20 random points $\subseteq [-1, 1]$ |
| f_2 -Benchmark | $x^5 + x^4 + x^3 + x^2 + x$ | 20 random points $\subseteq [-1, 1]$ |
| f_3 -Benchmark | $\sin(x^2) * \cos(x) - 1$ | 20 random points $\subseteq [-1, 1]$ |
| f_4 -Benchmark | $\log(x+1) + \log(x^2+1)$ | 20 random points $\subseteq [0, 2]$ |
| f_5 -Benchmark | $2\sin(x) * \cos(y)$ | 100 random points $\subseteq [-1, 1] \times [-1, 1]$ |

Table 2.4: Table with the parameters for GP used for the benchmark symbolic regression problems.

| Parameter | Description |
|-------------------------------|--|
| <i>Population size</i> | 200 individuals |
| <i>Generations</i> | 100 generations |
| <i>Initialization</i> | <i>Ramped Half-and-Half</i> , with 6 levels of maximum depth |
| <i>Operator probabilities</i> | Crossover $p_c = 0.9$, Mutation $p_\mu = 0.05$ |
| <i>Function set</i> | $(+, -, \times, \div, \sin, \cos, \exp, \log)$. |
| <i>Terminal set</i> | $v, 1$ for single variable problems and v, v for bivariable problem. |
| <i>Maximum tree depth</i> | 20 levels |
| <i>Selection</i> | Tournament selection of size 3 |
| <i>Elitism</i> | Best individual always survives |

are characterized by a high-dimensionality and a difficult to model behavior. The experimental parameters used are the same as those in Gonçalves and Silva (2013), summarized in Table 2.5. Fitness is calculated as the root mean square error between predicted and expected outputs, and the data set is randomly divided before each run, using 50% for training and 50% for testing.

For the benchmark problems, the five box plots in Figure 2.7 show the performance of the best individual from each run evaluated with the test set. Figure 2.8, shows the overfitting results for each method,

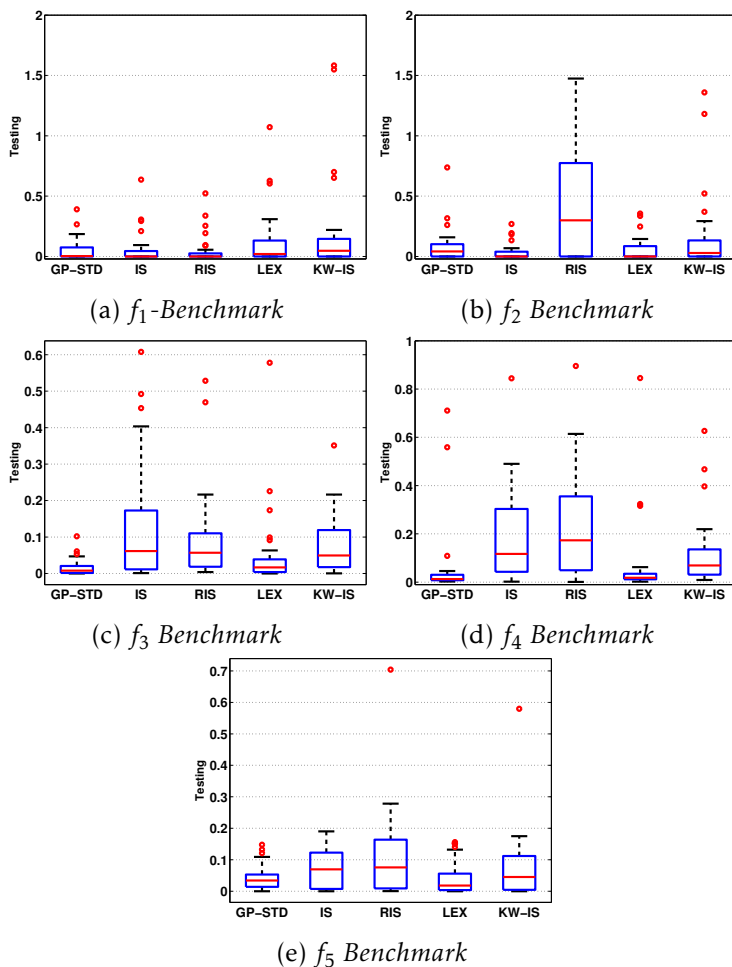


Figure 2.7: Box plot comparison about the test performance of the methods, from the best solution found for each benchmark symbolic regression problem over all thirty runs.

and Figure 2.9 summarizes the effect on bloat for each method, captured by the average size of the population in the final generation. Similar plots are shown for the real-world problems in Figures 2.10, 2.11

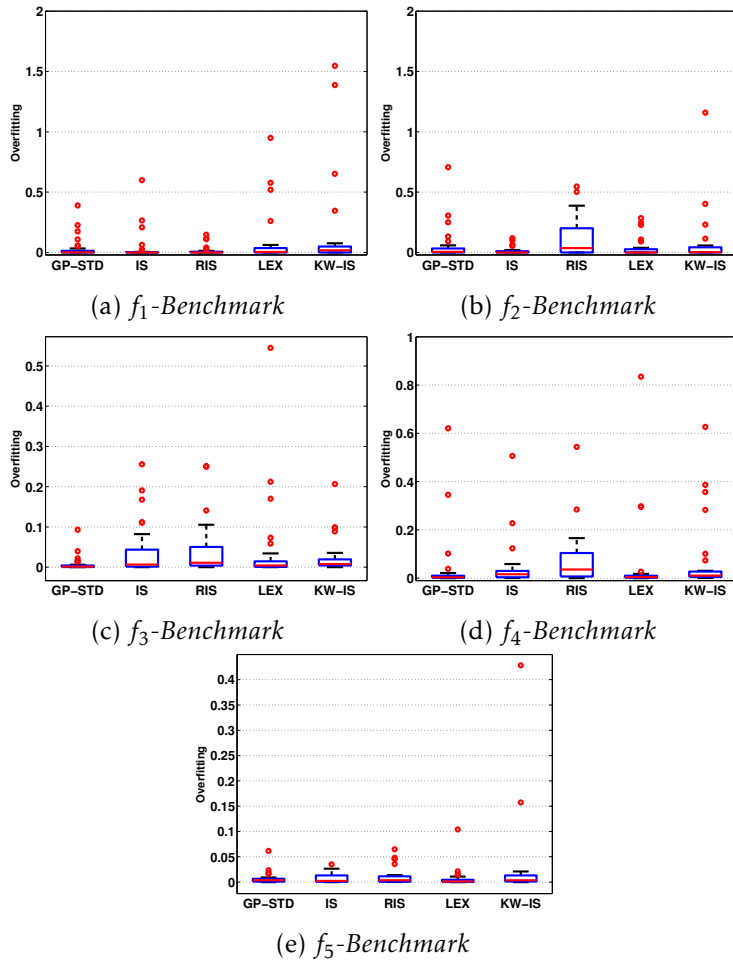


Figure 2.8: Box plot comparison about the overfitting performance of the methods, from the best solution found for each benchmark symbolic regression problem over all thirty runs.

and 2.12, for test performance, overfitting and average population size respectively.

To summarize these results a numerical comparison of the methods is provided in Table 2.6, showing the median values of each method,

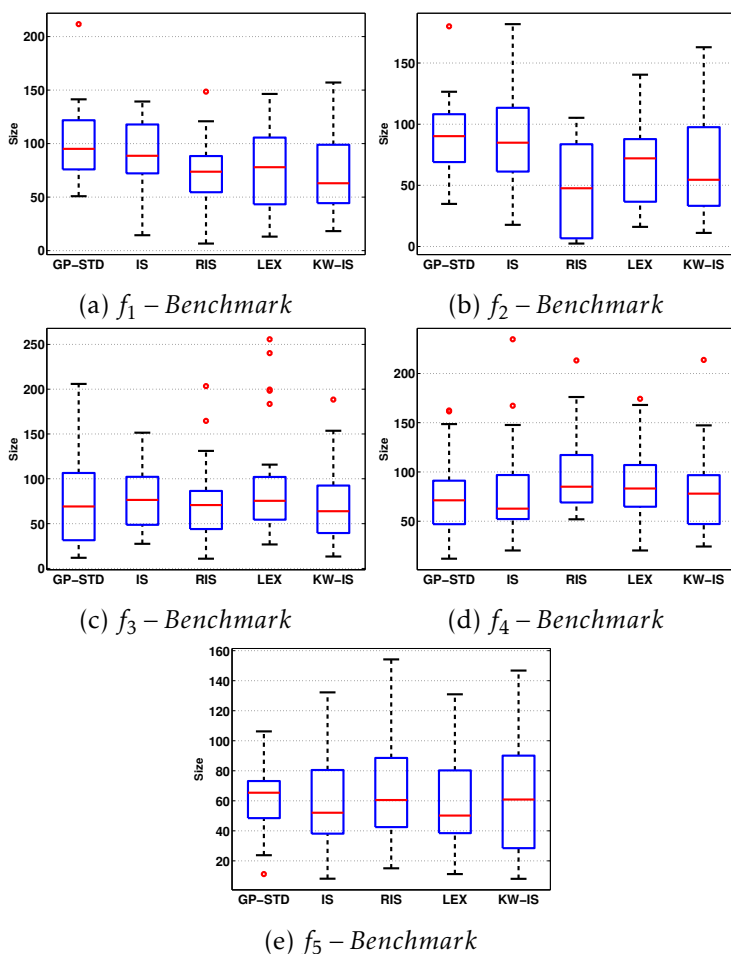


Figure 2.9: Box plot comparison about the average size performance of the methods, from the solutions found for each benchmark symbolic regression problem over all thirty runs.

on each problem and for each measure. Here, bold indicates statistically different results with respect to the other (non-bold) values. Table 2.7, shows the p-values with the Bonferroni-Holm correction for the pairwise comparisons on each problem where bold values indicate that

Table 2.5: Table with the parameters for GP used for symbolic regression real-world problems.

| Parameter | Description |
|-------------------------------|---|
| <i>Population size</i> | 500 individuals |
| <i>Generations</i> | 200 generations |
| <i>Initialization</i> | <i>Ramped Half-and-Half</i> , with 6 levels of maximum depth |
| <i>Operator probabilities</i> | Crossover $p_c = 0.9$, Mutation $p_\mu = 0.05$ |
| <i>Function set</i> | {+, -, ×, ÷} |
| <i>Terminal set</i> | Input variables, constants -1.0, -0.5, 0, 0.5 and 1.0 |
| <i>Maximum tree depth</i> | 17 levels |
| <i>Selection</i> | Tournament selection of size 10 |
| <i>Elitism</i> | Best individual always survives |

the null hypothesis is rejected at the $\alpha = 0.05$ significance level. The results on symbolic regression reveal several interesting trends.

First, considering test performance the following can be observed. For all benchmark problems none of the fitness-case sampling methods outperform GP-STD. In fact, only LEX can compare on test performance, while RIS achieves the worst results of all methods. On the other hand, for the real-world problems GP-STD is outperformed by at least 2 fitness-case sampling methods, and in two problems by six of them. In these problems, LEX and KW-IS clearly show the best performance, while IS outperforms GP-STD on two of the six problems (Toxicity and Plasma). Again, RIS clearly is the worst of all fitness-case sampling methods.

Based on overfitting, IS shows the best performance, on both the synthetic benchmarks and the real-world problems. Similarly, RIS shows strong performance on the real-world cases. The performance of IS and RIS is consistent with those reported in Gonçalves and Silva (2013). On the other hand, GP-STD and LEX do not seem to overfit on synthetic problems, but clearly do so on the real-world cases. In particular, GP-STD clearly shows the worst performance among all methods

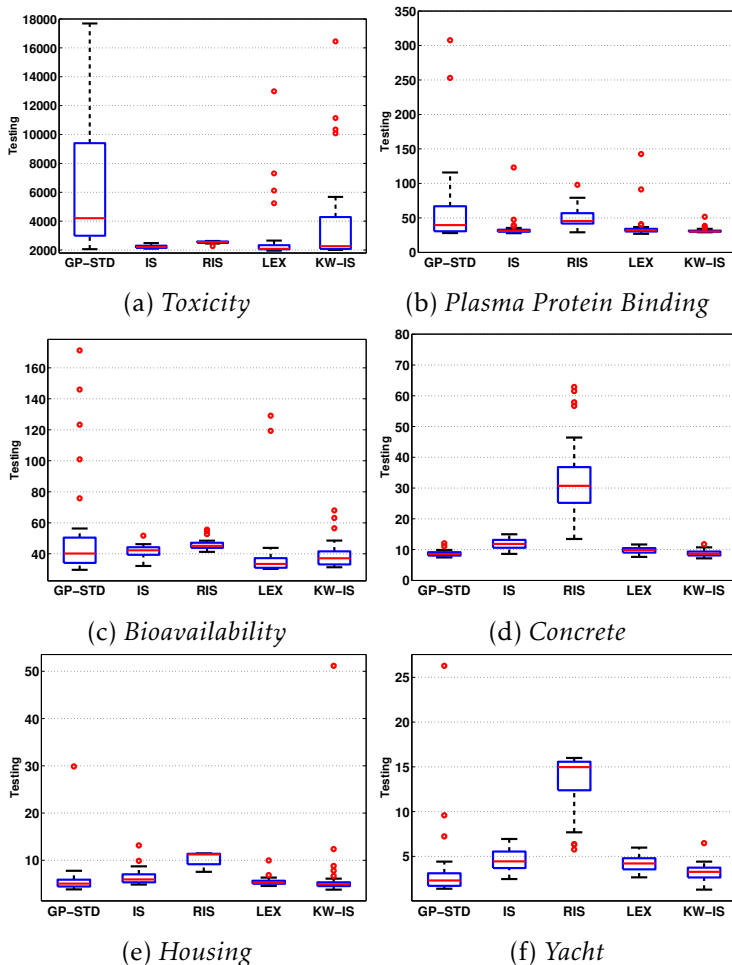


Figure 2.10: Box plot comparison about the test performance of the methods, from the best solution found for each real-world regression problem over all thirty runs.

on the real-world problems. KW-IS shows the weakest performance among all fitness-case sampling methods.

Finally if we consider size, some interesting results can be seen. On synthetic problems there appears to be small differences between the

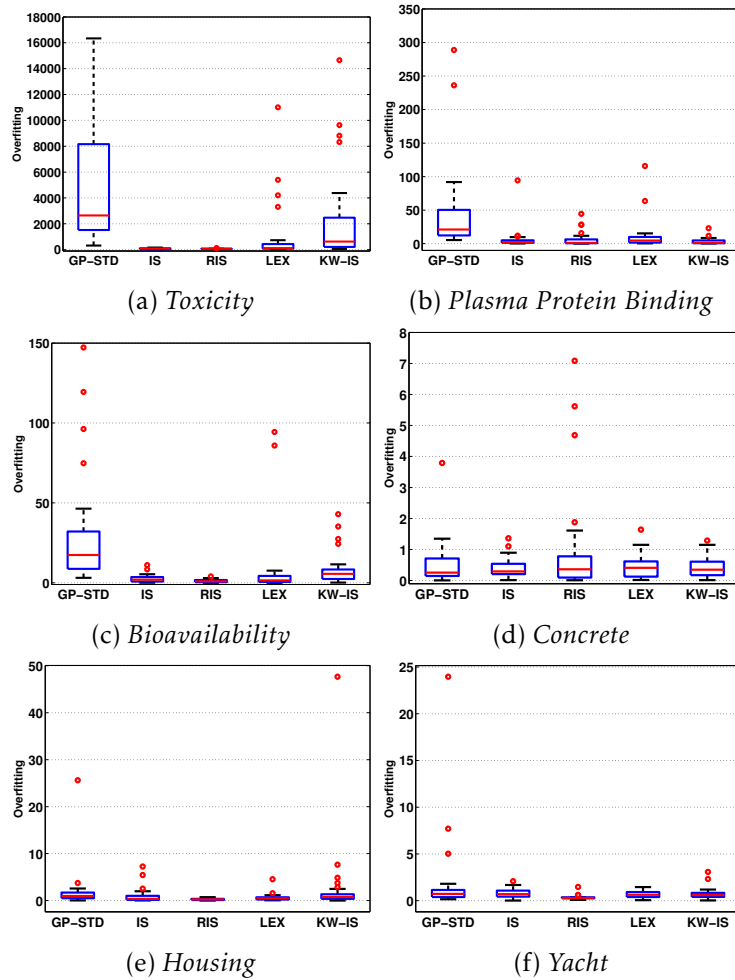


Figure 2.11: Box plot comparison about the overfitting performance of the methods, from the best solution found for each real-world regression problem over all thirty runs.

methods, where only can see difference statistically significant in the second problem. However, on the real-world cases more interesting results are obtained. First, the reason for the bad test performance of RIS can be attributed to the fact that the evolved trees are basically a single

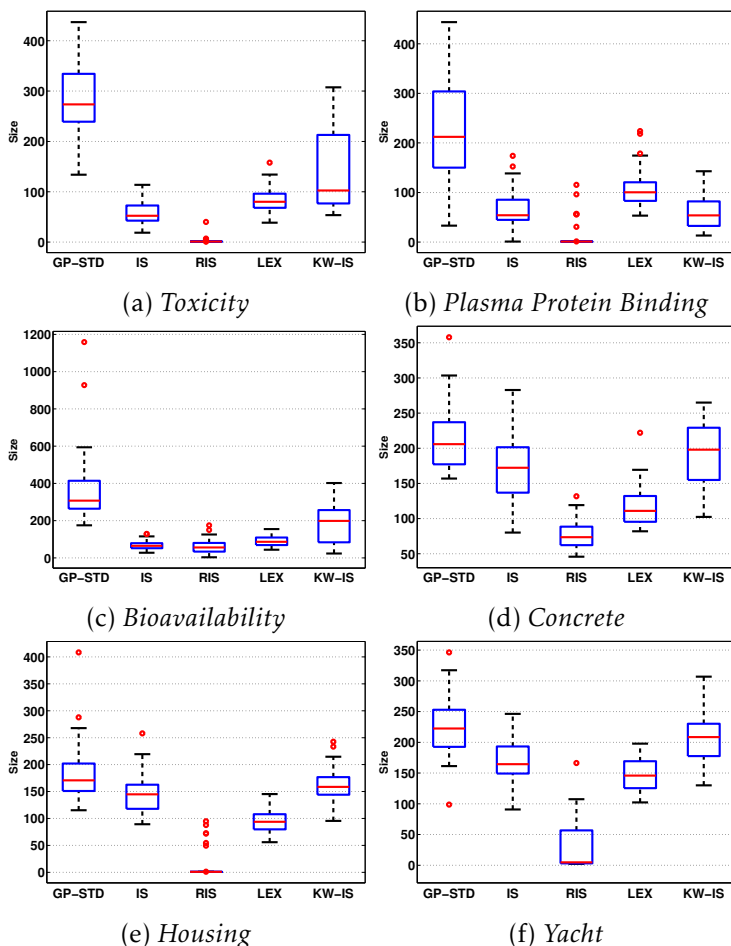


Figure 2.12: Box plot comparison about the average size performance of the methods, from the solutions found for each real-world regression problem over all thirty runs.

terminal (variable) in three of the six cases. Second, GP-STD consistently produces the larger trees, in some cases one order of magnitude larger than the fitness-case sampling methods. Third, if we consider size and test fitness, then we can say that IS, LEX and KW-IS produce smaller trees with better performance than GP-STD.

Table 2.6: Table show the median of 30 executions for testing, overfitting and size; bold indicates best.

| <i>Testing</i> | | | | | |
|-------------------------------|---------------|----------------|---------------|----------------|----------------|
| | GP-STD | IS | RIS | LEX | KW-IS |
| <i>f1-Benchmark</i> | 0.0031 | 0.0000 | 0.0000 | 0.0207 | 0.0468 |
| <i>f2-Benchmark</i> | 0.0418 | 0.0000 | 0.2990 | 0.0000 | 0.0286 |
| <i>f3-Benchmark</i> | 0.0079 | 0.0616 | 0.0569 | 0.0166 | 0.0496 |
| <i>f4-Benchmark</i> | 0.0126 | 0.1172 | 0.1733 | 0.0180 | 0.0691 |
| <i>f5-Benchmark</i> | 0.0342 | 0.0695 | 0.0756 | 0.0180 | 0.0455 |
| <i>Toxicity</i> | 4206.99 | 2217.21 | 2555.48 | 2089.20 | 2267.04 |
| <i>Plasma Protein Binding</i> | 39.47 | 31.21 | 45.47 | 32.00 | 30.31 |
| <i>Bioavailability</i> | 40.16 | 42.23 | 45.15 | 33.49 | 37.12 |
| <i>Concrete</i> | 8.56 | 11.84 | 30.72 | 9.87 | 8.67 |
| <i>Housing</i> | 5.08 | 5.96 | 11.26 | 5.26 | 4.93 |
| <i>Yacht</i> | 2.31 | 4.45 | 14.96 | 4.21 | 3.27 |
| <i>Overfitting</i> | | | | | |
| <i>f1-Benchmark</i> | 0.0010 | 0.0000 | 0.0000 | 0.0039 | 0.0184 |
| <i>f2-Benchmark</i> | 0.0052 | 0.0000 | 0.0366 | 0.0000 | 0.0023 |
| <i>f3-Benchmark</i> | 0.0013 | 0.0065 | 0.0111 | 0.0044 | 0.0078 |
| <i>f4-Benchmark</i> | 0.0039 | 0.0159 | 0.0362 | 0.0041 | 0.0109 |
| <i>f5-Benchmark</i> | 0.0036 | 0.0025 | 0.0036 | 0.0011 | 0.0036 |
| <i>Toxicity</i> | 2645.66 | 81.56 | 65.03 | 115.32 | 629.16 |
| <i>Plasma Protein Binding</i> | 21.19 | 2.08 | 1.21 | 4.67 | 1.25 |
| <i>Bioavailability</i> | 17.40 | 2.01 | 0.94 | 1.50 | 5.64 |
| <i>Concrete</i> | 0.2541 | 0.2950 | 0.3609 | 0.4100 | 0.3486 |
| <i>Housing</i> | 0.9882 | 0.3800 | 0.2519 | 0.4918 | 0.7668 |
| <i>Yacht</i> | 0.7244 | 0.6957 | 0.2884 | 0.6293 | 0.5997 |
| <i>Size</i> | | | | | |
| <i>f1-Benchmark</i> | 95 | 89 | 74 | 78 | 63 |
| <i>f2-Benchmark</i> | 90 | 85 | 48 | 72 | 55 |
| <i>f3-Benchmark</i> | 69 | 76 | 71 | 75 | 64 |
| <i>f4-Benchmark</i> | 71 | 63 | 85 | 83 | 78 |
| <i>f5-Benchmark</i> | 65 | 52 | 61 | 50 | 61 |
| <i>Toxicity</i> | 274 | 52 | 1 | 80 | 102 |
| <i>Plasma Protein Binding</i> | 212 | 54 | 1 | 100 | 54 |
| <i>Bioavailability</i> | 308 | 65 | 57 | 87 | 199 |
| <i>Concrete</i> | 206 | 172 | 74 | 111 | 198 |
| <i>Housing</i> | 171 | 145 | 1 | 94 | 159 |
| <i>Yacht</i> | 222 | 164 | 5 | 146 | 209 |

2.6.4.2 Classification Problems

Synthetic binary classification problems were randomly generated using Gaussian mixture models (GMM's). Examples of the classifica-

GENETIC PROGRAMMING

Table 2.7: Results of the Friedman test for the classification problems, showing the p-value after the Bonferroni-Holm correction for each pairwise comparison; bold indicates that the test rejects the null hypothesis at the $\alpha = 0.05$ significance level.

| | Testing | | | | | Overfitting | | | | | Size | | | | |
|-------------------------------|---------|--------|--------|--------|--------|-------------|--------|--------|--------|--------|--------|--------|--------|--------|--|
| | GP-STD | IS | LEX | KW-IS | GP-STD | IS | RIS | LEX | KW-IS | GP-STD | IS | RIS | LEX | KW-IS | |
| <i>f1-Benchmark</i> | GP-STD | 1.5948 | 1.2921 | 1.7988 | 2.3515 | 0.5431 | 0.8648 | 1.2842 | 1.4126 | 2.0000 | 0.5431 | 1.0089 | 0.2561 | 1.0089 | |
| | IS | 1.3662 | 2.3515 | 0.1674 | 1.4300 | 0.5431 | 0.1674 | 1.4300 | 0.0250 | 2.0000 | 0.5431 | 1.0089 | 0.8648 | 1.0089 | |
| | RIS | 1.7988 | 0.1674 | 1.4300 | 0.0250 | 1.4300 | 0.1674 | 1.4300 | 0.0250 | 2.0000 | 0.5431 | 1.0089 | 0.8648 | 1.0089 | |
| | LEX | 2.1190 | 1.4300 | 0.0250 | 1.4300 | 0.1674 | 1.4300 | 0.0250 | 1.4300 | 0.0250 | 2.0000 | 0.5431 | 1.0089 | 0.8648 | |
| | KW-IS | 0.0481 | 0.0847 | 0.3527 | 1.0000 | 0.0314 | 0.9519 | 0.1867 | 1.0933 | 0.9304 | 0.0102 | 0.5431 | 0.5431 | 1.3956 | |
| <i>f2-Benchmark</i> | IS | 0.0067 | 0.3809 | 0.3593 | 0.5431 | 0.0016 | 0.4073 | 1.0933 | 0.9304 | 0.0102 | 0.5431 | 0.7206 | 0.7206 | 0.7206 | |
| | RIS | 0.2876 | 0.3787 | 0.7063 | 0.4073 | 0.3787 | 0.7063 | 0.9304 | 0.9304 | 0.0102 | 0.5431 | 0.9304 | 0.4752 | 0.8200 | |
| | LEX | 0.7063 | 0.9304 | 0.9304 | 0.9304 | 0.9304 | 0.9304 | 0.9304 | 0.9304 | 0.0102 | 0.5431 | 0.9304 | 0.9304 | 0.9304 | |
| | KW-IS | 0.0279 | 0.0091 | 0.0006 | 0.0006 | 0.0847 | 0.0102 | 1.0089 | 0.0314 | 3.5750 | 1.9133 | 3.5750 | 2.8600 | 2.8600 | |
| | GP-STD | 0.9304 | 0.4073 | 0.9304 | 0.0089 | 1.0933 | 0.9304 | 0.0089 | 1.0933 | 2.7913 | 2.1450 | 1.4413 | 1.4413 | 1.4413 | |
| <i>f3-Benchmark</i> | IS | 0.0741 | 0.8200 | 0.8648 | 1.0933 | 0.8648 | 1.0933 | 0.8648 | 1.0933 | 2.7913 | 2.1450 | 1.4413 | 1.4300 | 1.4300 | |
| | RIS | 0.7206 | 0.9304 | 0.9304 | 0.9304 | 0.9304 | 0.9304 | 0.9304 | 0.9304 | 2.7913 | 2.1450 | 1.4413 | 1.4300 | 1.4300 | |
| | LEX | 0.9304 | 0.9304 | 0.9304 | 0.9304 | 0.9304 | 0.9304 | 0.9304 | 0.9304 | 2.7913 | 2.1450 | 1.4413 | 1.4300 | 1.4300 | |
| | KW-IS | 0.0244 | 0.0026 | 0.5466 | 0.0244 | 0.4752 | 0.0314 | 0.9304 | 0.8200 | 2.1450 | 1.2971 | 1.2971 | 2.8600 | 2.8600 | |
| | GP-STD | 0.5765 | 0.0209 | 0.5765 | 0.0209 | 0.7206 | 0.4752 | 0.9304 | 0.8200 | 1.3666 | 1.1530 | 2.8600 | 2.8600 | 2.8600 | |
| <i>f4-Benchmark</i> | IS | 0.0209 | 0.5765 | 0.0209 | 0.0209 | 0.0209 | 0.5765 | 0.0209 | 0.0209 | 2.0000 | 0.6789 | 2.0000 | 0.6789 | 2.0000 | |
| | RIS | 0.0209 | 0.0209 | 0.0209 | 0.0209 | 0.0209 | 0.0209 | 0.0209 | 0.0209 | 2.0000 | 0.6789 | 2.0000 | 0.6789 | 2.0000 | |
| | LEX | 0.0209 | 0.0209 | 0.0209 | 0.0209 | 0.0209 | 0.0209 | 0.0209 | 0.0209 | 2.0000 | 0.6789 | 2.0000 | 0.6789 | 2.0000 | |
| | KW-IS | 0.0209 | 0.0209 | 0.0209 | 0.0209 | 0.0209 | 0.0209 | 0.0209 | 0.0209 | 2.0000 | 0.6789 | 2.0000 | 0.6789 | 2.0000 | |
| | GP-STD | 2.1866 | 0.6789 | 1.4300 | 1.7428 | 2.3260 | 2.1450 | 1.2971 | 2.1190 | 2.1450 | 2.0000 | 2.4599 | 4.2900 | 4.2900 | |
| <i>f5-Benchmark</i> | IS | 1.4300 | 2.1866 | 2.3260 | 2.3260 | 2.3260 | 2.3260 | 2.3260 | 2.1450 | 2.1450 | 2.4599 | 4.2900 | 1.4413 | 1.4413 | |
| | RIS | 2.3260 | 1.9133 | 1.8668 | 1.8668 | 1.8668 | 1.8668 | 1.8668 | 2.1866 | 2.1866 | 2.4599 | 4.2900 | 1.4413 | 1.4413 | |
| | LEX | 1.8668 | 1.8668 | 1.8668 | 1.8668 | 1.8668 | 1.8668 | 1.8668 | 2.1866 | 2.1866 | 2.4599 | 4.2900 | 1.4413 | 1.4413 | |
| | KW-IS | 0.0001 | 0.0005 | 0.0209 | 0.0244 | 0.0000 | 0.0000 | 0.0004 | 0.0318 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| | GP-STD | 0.0529 | 0.7150 | 0.1358 | 0.0000 | 0.7150 | 0.1358 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| <i>Toxicity</i> | IS | 0.0000 | 1.4300 | 1.4300 | 1.4300 | 0.4324 | 0.1423 | 0.9304 | 0.4324 | 0.0003 | 0.0020 | 0.7150 | 0.7150 | 0.7150 | |
| | RIS | 0.0005 | 0.0000 | 0.1708 | 0.1708 | 0.2716 | 0.9304 | 0.2716 | 0.9304 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| | LEX | 0.1708 | 0.1708 | 0.1708 | 0.1708 | 0.2716 | 0.9304 | 0.2716 | 0.9304 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| | KW-IS | 0.4652 | 0.4324 | 0.3394 | 0.4324 | 0.0000 | 0.0000 | 0.0001 | 0.0001 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| | GP-STD | 0.0081 | 0.0081 | 0.0209 | 0.0209 | 0.1138 | 0.2883 | 0.0013 | 0.0013 | 0.1138 | 0.1138 | 0.1138 | 0.1138 | 0.1138 | |
| <i>Plasma Protein Binding</i> | IS | 0.0000 | 0.0000 | 0.0023 | 0.0023 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| | RIS | 0.0000 | 0.0000 | 0.0023 | 0.0023 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| | LEX | 0.0000 | 0.0000 | 0.0023 | 0.0023 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| | KW-IS | 0.0000 | 0.0000 | 0.0023 | 0.0023 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| | GP-STD | 0.4652 | 0.4324 | 0.3394 | 0.4324 | 0.0000 | 0.0000 | 0.0001 | 0.0001 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| <i>Bioavailability</i> | IS | 0.0081 | 0.0081 | 0.0209 | 0.0209 | 0.1138 | 0.2883 | 0.0013 | 0.0013 | 0.1138 | 0.1138 | 0.1138 | 0.1138 | 0.1138 | |
| | RIS | 0.0000 | 0.0000 | 0.0023 | 0.0023 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| | LEX | 0.0000 | 0.0000 | 0.0023 | 0.0023 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| | KW-IS | 0.0000 | 0.0000 | 0.0023 | 0.0023 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| | GP-STD | 0.0000 | 0.0105 | 1.0000 | 1.0000 | 3.2565 | 1.4413 | 1.4300 | 2.8600 | 0.0318 | 0.0000 | 0.0000 | 0.1441 | 0.1441 | |
| <i>Concrete</i> | IS | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 3.2565 | 2.4599 | 2.8600 | 2.8600 | 0.0000 | 0.0000 | 0.0000 | 0.0569 | 0.0569 | |
| | RIS | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 2.4599 | 2.8600 | 2.8600 | 2.8600 | 0.0000 | 0.0000 | 0.0000 | 0.0569 | 0.0569 | |
| | LEX | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 2.4599 | 2.8600 | 2.8600 | 2.8600 | 0.0000 | 0.0000 | 0.0000 | 0.0569 | 0.0569 | |
| | KW-IS | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 2.4599 | 2.8600 | 2.8600 | 2.8600 | 0.0000 | 0.0000 | 0.0000 | 0.0569 | 0.0569 | |
| | GP-STD | 0.0529 | 0.0000 | 1.4300 | 1.4300 | 0.8648 | 0.0001 | 0.847 | 1.3956 | 0.0030 | 0.0000 | 0.0000 | 0.1358 | 0.1358 | |
| <i>Housing</i> | IS | 0.0004 | 0.0529 | 0.0209 | 0.0209 | 1.3956 | 0.9304 | 0.8648 | 0.8648 | 0.0000 | 0.0002 | 0.4652 | 0.4652 | 0.4652 | |
| | RIS | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.9304 | 0.8648 | 0.8648 | 0.8648 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| | LEX | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.9304 | 0.8648 | 0.8648 | 0.8648 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| | KW-IS | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.9304 | 0.8648 | 0.8648 | 0.8648 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| | GP-STD | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.9304 | 0.8648 | 0.8648 | 0.8648 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| <i>Yacht</i> | IS | 0.0004 | 0.0000 | 0.0010 | 0.1358 | 2.1450 | 0.0023 | 2.7913 | 2.0000 | 0.0001 | 0.0000 | 0.0000 | 0.7150 | 0.7150 | |
| | RIS | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 2.7913 | 2.8600 | 2.8600 | 2.8600 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| | LEX | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 2.7913 | 2.8600 | 2.8600 | 2.8600 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| | KW-IS | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 2.7913 | 2.8600 | 2.8600 | 2.8600 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |
| | GP-STD | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 2.7913 | 2.8600 | 2.8600 | 2.8600 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | |

tion problems generated showed in Figure 2.13; which depicts sample points of two different classes (circles and crosses) scattered over the \mathbb{R}^2 plane. Problems are generated with unimodal or multimodal classes, with different amounts of class overlap. All class samples lie within the closed 2-D interval $v_1, v_2 \in [-10, 10]$, and 200 sample points were randomly generated for each class. The parameters for the GMM of each class were also randomly chosen using the following ranges of values:

1. Number of Gaussian components: $\{1, 2, 3\}$.
2. Median of each Gaussian component for each feature dimension: $[-3, 3]$.
3. Each element of the covariant matrix of each Gaussian component: $(0, 2]$.
4. The rotation angle of each covariance matrix: $[0, 2\pi]$.
5. The proportion of sample points generated with each Gaussian component: $[0, 1]$.

Afterwards, five problem were chosen, showed in Figure 2.13, that are used to evaluate the algorithms tested here. Additionally, six real-world problems from the University of California, Irvine (UCI) machine learning repository were chosen (Lichman, 2013), summarized in Table 5.5. These problems have a more complex nature and therefore will be interesting test cases for the sampling methods

In this work, was used a GP classifier called static range selection (SRS) (Zhang and Smart, 2006). For a two class problem and real-valued GP outputs, the SRS decision rule is simple: if the program output for input pattern v is greater than zero then the pattern is labeled as belonging to class A, otherwise its labeled as a class B pattern. The experimental parameters are given in Table 2.9, and fitness is given by the classification error.

For the synthetic problems, Figures 2.14, 2.15 and 2.16 illustrate the performance of each method on test fitness, overfitting and average size. Similarly, Figures 2.17, 2.18 and 2.19 show the same information for the real-world problems.

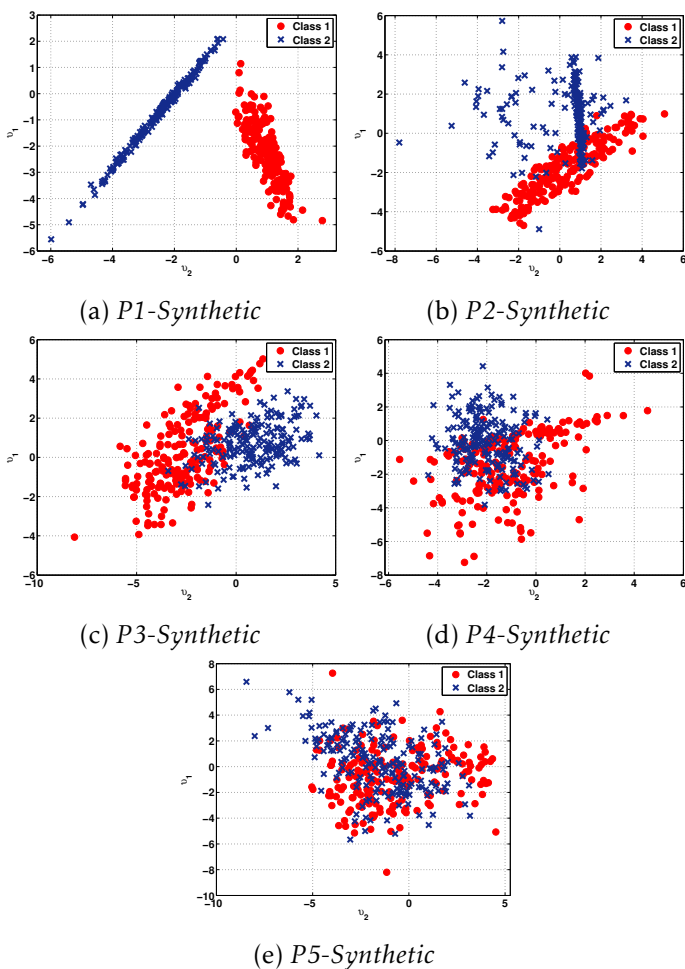


Figure 2.13: Synthetic binary classification problems randomly generated using Gaussian mixture models with different amounts of class overlap, scattered over 2 dimensional space.

To summarize these results a numerical comparison of the methods is provided in Table 2.10, showing the median values of each method, on each problem and for each measure. Here, bold indicates statistically different results with respect to the other (non-bold) values. Table

Table 2.8: Real-world classification problems from Irvine (UCI) machine learning repository.

| No. | Problem | Classes | Features | Instances | Description |
|-----|--------------------------------|---------|----------|-----------|--|
| 1 | <i>Breast Cancer Wisconsin</i> | 2 | 8 | 699 | Original Wisconsin Breast Cancer Database. |
| 2 | <i>Parkinson's</i> | 2 | 22 | 195 | Oxford Parkinson's Disease Detection Dataset |
| 3 | <i>Pima Indians Diabetes</i> | 2 | 8 | 768 | From National Institute of Diabetes |
| 4 | <i>Indian Liver Patient</i> | 2 | 10 | 579 | Contains 416 liver patient records and 167 non liver patient records |
| 5 | <i>Retinopathy</i> | 2 | 19 | 1151 | Contains features extracted from the Messidor image set, to predict signs of diabetic retinopathy or not |
| 6 | <i>Vertebral Column 2C</i> | 2 | 6 | 310 | Biomedical data set built by Dr. Henrique da Mota |

Table 2.9: Table with the parameters for GP used for the classification problems.

| Parameter | Description |
|-------------------------------|---|
| <i>Runs</i> | 30 |
| <i>Size of population</i> | 200 individuals |
| <i>Generations</i> | 100 generations |
| <i>Initialization</i> | <i>Ramped Half-and-half</i> with maximum depth level of 6 |
| <i>Operator probabilities</i> | Crossover $p_c = 0.8$, mutation $p_\mu = 0.2$ |
| <i>Function set</i> | $(+, -, \times, \div, \sin, \cos, \exp, \log, if)$ |
| <i>Terminal set</i> | input variables |
| <i>Maximum tree depth</i> | 20 levels |
| <i>Selection</i> | Size 3 tournament |
| <i>Elitism</i> | Best individual always survives |

2.11, shows the p-values with the Bonferroni-Holm correction for the pairwise comparisons on each problem where bold values indicate that the null hypothesis is rejected at the $\alpha = 0.05$ significance level.

Based on test performance, three algorithms consistently show the best results, GP-STD, LEX and KW-IS. On the other hand, RIS and IS are clearly the worst methods, in some cases their performance shows twice the error as the best methods (*Breast Cancer Wisconsin* and *P2-*

GENETIC PROGRAMMING

Table 2.10: Table show the median of 30 executions over testing, overfitting and size; bold indicates best.

| <i>Testing</i> | | | | | |
|--------------------------------|---------------|---------------|---------------|---------------|---------------|
| | GP-STD | IS | RIS | LEX | KW-IS |
| <i>P1-Synthetic</i> | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| <i>P2-Synthetic</i> | 0.0938 | 0.1937 | 0.1750 | 0.1000 | 0.1000 |
| <i>P3-Synthetic</i> | 0.1375 | 0.2062 | 0.1875 | 0.1375 | 0.1375 |
| <i>P4-Synthetic</i> | 0.2750 | 0.2500 | 0.3000 | 0.2625 | 0.2750 |
| <i>P5-Synthetic</i> | 0.3875 | 0.3750 | 0.3438 | 0.3688 | 0.3875 |
| <i>Breast Cancer Wisconsin</i> | 0.1023 | 0.2045 | 0.1932 | 0.0795 | 0.0909 |
| <i>Parkinson's</i> | 0.1795 | 0.2564 | 0.2564 | 0.1538 | 0.2308 |
| <i>Pima Indians Diabetes</i> | 0.2727 | 0.3506 | 0.3506 | 0.2630 | 0.2955 |
| <i>Indian Liver Patient</i> | 0.2832 | 0.2920 | 0.2832 | 0.2832 | 0.2832 |
| <i>Retinopathy</i> | 0.2860 | 0.3362 | 0.4214 | 0.3166 | 0.2729 |
| <i>Vertebral Column 2C</i> | 0.1855 | 0.2258 | 0.2258 | 0.2097 | 0.2258 |
| <i>Overfitting</i> | | | | | |
| <i>P1-Synthetic</i> | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| <i>P2-Synthetic</i> | 0.0359 | 0.0547 | 0.0344 | 0.0328 | 0.0203 |
| <i>P3-Synthetic</i> | 0.0469 | 0.0312 | 0.0312 | 0.0500 | 0.0281 |
| <i>P4-Synthetic</i> | 0.0859 | 0.0344 | 0.0344 | 0.0625 | 0.0375 |
| <i>P5-Synthetic</i> | 0.1063 | 0.0656 | 0.0516 | 0.0859 | 0.0609 |
| <i>Breast Cancer Wisconsin</i> | 0.0428 | 0.0360 | 0.0335 | 0.0312 | 0.0454 |
| <i>Parkinson's</i> | 0.0385 | 0.0128 | 0.0128 | 0.0385 | 0.0321 |
| <i>Pima Indians Diabetes</i> | 0.0398 | 0.0021 | 0.0081 | 0.0291 | 0.0219 |
| <i>Indian Liver Patient</i> | 0.0536 | 0.0216 | 0.0038 | 0.0082 | 0.0077 |
| <i>Retinopathy</i> | 0.0401 | 0.0241 | 0.0214 | 0.0245 | 0.0215 |
| <i>Vertebral Column 2C</i> | 0.0565 | 0.0645 | 0.0444 | 0.0484 | 0.0504 |
| <i>Size</i> | | | | | |
| <i>P1-Synthetic</i> | 149 | 147 | 124 | 20 | 143 |
| <i>P2-Synthetic</i> | 91 | 5 | 6 | 102 | 107 |
| <i>P3-Synthetic</i> | 82 | 6 | 5 | 94 | 103 |
| <i>P4-Synthetic</i> | 210 | 168 | 100 | 90 | 71 |
| <i>P5-Synthetic</i> | 215 | 162 | 78 | 128 | 66 |
| <i>Breast Cancer Wisconsin</i> | 50 | 3 | 4 | 41 | 59 |
| <i>Parkinson's</i> | 16 | 2 | 3 | 17 | 48 |
| <i>Pima Indians Diabetes</i> | 70 | 4 | 3 | 66 | 13 |
| <i>Indian Liver Patient</i> | 165 | 108 | 65 | 79 | 57 |
| <i>Retinopathy</i> | 153 | 103 | 4 | 61 | 5 |
| <i>Vertebral Column 2C</i> | 162 | 118 | 88 | 112 | 51 |

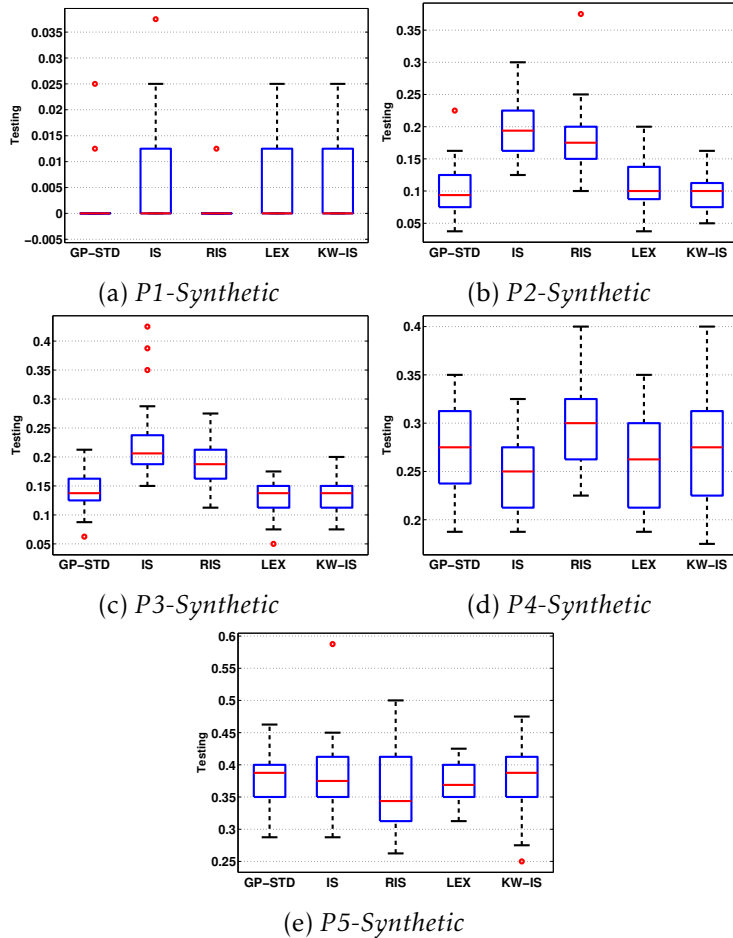


Figure 2.14: Box plot comparison about the test performance of the methods, from the best solution found for each synthetic classification problem over all thirty run.

Synthetic). Therefore, these results suggest that no performance improvement is obtained by using fitness-case sampling methods. Based on overfitting, again IS and RIS show the best results, however given their worse performance based on test fitness, they should not be preferred. Among the other methods, the best overfitting performance was

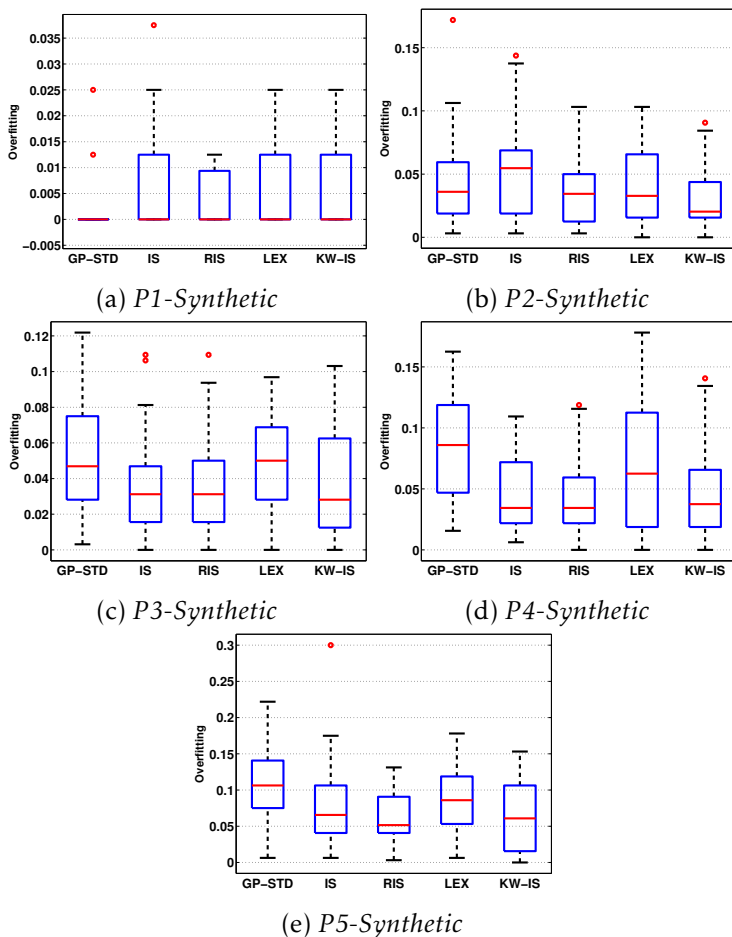


Figure 2.15: Box plot comparison about the overfitting performance of the methods, from the best solution found for each synthetic classification problem over all thirty run.

exhibited by KW-IS, while GP-STD showed the worst. Finally, based on size, again IS and RIS evolve the smaller trees, but the improvement in program size is not justified by their poor performance. On the other hand, LEX and KW-IS sometimes evolve smaller trees than GP-STD, but in other cases their performance is similar or a bit worse. Given

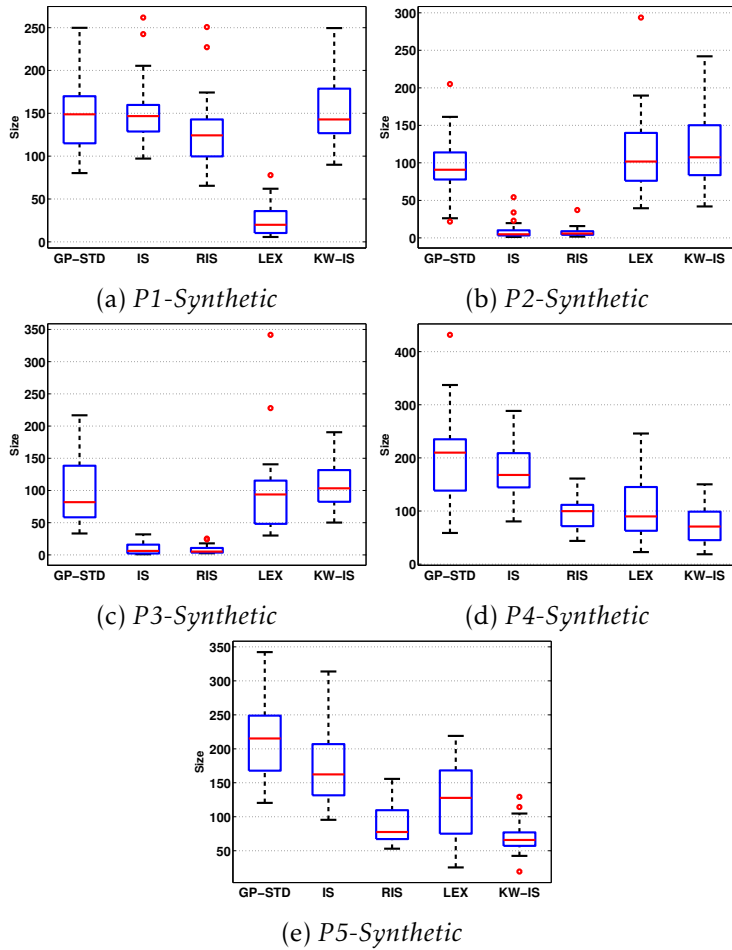


Figure 2.16: Box plot comparison about the average size performance of the methods, from the solutions found for each synthetic classification problem over all thirty run.

these results, it can be stated that the LEX and KW-IS do not improve upon GP-STD, but they do not compromise performance either, while IS and RIS negatively effect classifier performance.

GENETIC PROGRAMMING

Table 2.11: Results of the Friedman test for the classification problems, showing the p-value after the Bonferroni-Holm correction for each pairwise comparison; bold indicates that the test rejects the null hypothesis at the $\alpha = 0.05$ significance level.

| | Testing | | | | | Overfitting | | | | | Size | | | | |
|--------------------------------|---------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|--------|---------------|---------------|---------------|---------------|--|
| | GP-STD | IS | LEX | KW-IS | GP-STD | IS | RIS | LEX | KW-IS | GP-STD | IS | RIS | LEX | KW-IS | |
| <i>P1-Synthetic</i> | GP-STD | 0.7494 | 1.8512 | 2.0269 | 1.3515 | 0.8326 | 1.8674 | 2.4323 | 1.8020 | - | 2.1450 | 0.0000 | 0.0000 | 2.1450 | |
| | IS | - | 0.3251 | 1.8674 | 2.0269 | - | 1.2025 | 1.7789 | 2.1929 | - | 0.0209 | 0.0000 | 0.0000 | 2.0000 | |
| | LEX | - | - | 0.7494 | 1.6661 | - | 2.4323 | 1.9953 | - | - | - | - | 0.0000 | 0.1423 | |
| | KW-IS | - | - | - | 1.5260 | - | - | 1.2342 | - | - | - | - | - | 0.0000 | |
| | GP-STD | 0.0000 | 0.0003 | 1.0595 | 0.8655 | 0.2846 | 2.8600 | 2.8600 | 1.7054 | - | 0.0000 | 0.0000 | 0.5765 | 0.8200 | |
| <i>P2-Synthetic</i> | IS | - | 0.7117 | 0.0000 | 0.0000 | - | 0.6661 | 1.6399 | 0.6110 | - | 0.9304 | 0.0000 | 0.0000 | | |
| | RIS | - | - | 0.0000 | 0.0000 | - | - | 1.3555 | 2.1450 | - | - | 0.0000 | 0.0000 | | |
| | LEX | - | - | - | 1.0595 | - | - | 2.3260 | - | - | - | - | 0.9304 | | |
| | KW-IS | - | - | - | - | - | - | - | - | - | - | - | - | | |
| | GP-STD | 0.0001 | 0.0125 | 1.7324 | 1.7324 | 1.1530 | 0.6789 | 2.1450 | 0.6789 | - | 0.0000 | 0.0000 | 0.9304 | 0.2716 | |
| <i>P3-Synthetic</i> | IS | - | 0.7117 | 0.0000 | 0.0000 | - | 1.7054 | 1.1530 | 2.1450 | - | 1.0000 | 0.0000 | 0.0000 | | |
| | RIS | - | - | 0.0001 | - | - | 1.3666 | 1.8608 | - | - | - | 0.0000 | 0.0000 | | |
| | LEX | - | - | - | 1.1549 | - | - | 1.0089 | - | - | - | - | 0.2716 | | |
| | KW-IS | - | - | - | - | - | - | - | - | - | - | - | - | | |
| | GP-STD | 2.0155 | 0.9140 | 2.2548 | 1.1274 | 0.2561 | 0.0006 | 0.9682 | 0.2561 | - | 0.2716 | 0.0000 | 0.0013 | 0.0001 | |
| <i>P4-Synthetic</i> | IS | - | 0.0016 | 2.2548 | 1.6911 | - | 1.3956 | 1.3956 | 1.0933 | - | 0.0001 | 0.0004 | 0.0001 | | |
| | RIS | - | - | 0.0481 | 0.4703 | - | 0.4752 | 0.9304 | - | - | 1.0000 | 0.2716 | - | | |
| | LEX | - | - | - | 2.1638 | - | - | 0.8648 | - | - | - | - | 0.2037 | | |
| | KW-IS | - | - | - | - | - | - | - | - | - | - | - | - | | |
| | GP-STD | 2.8185 | 1.5492 | 1.5492 | 1.3057 | 0.4752 | 0.0349 | 1.3666 | 0.0953 | - | 0.0174 | 0.0000 | 0.0000 | | |
| <i>P5-Synthetic</i> | IS | - | 2.8185 | 1.7324 | 1.6948 | - | 1.3666 | 0.3680 | 1.1549 | - | 0.0000 | 0.0174 | 0.0000 | | |
| | RIS | - | - | 2.5966 | 1.3057 | - | 0.2277 | 1.3956 | - | - | 0.1441 | 0.1358 | - | | |
| | LEX | - | - | - | 2.3099 | - | - | 1.3956 | - | - | - | - | 0.0139 | | |
| | KW-IS | - | - | - | - | - | - | - | - | - | - | - | - | | |
| | GP-STD | 0.0000 | 0.0000 | 0.5223 | 0.8415 | 3.5750 | 3.0000 | 3.2565 | 2.7332 | - | 0.0000 | 0.0000 | 0.5466 | 0.5765 | |
| <i>Breast Cancer Wisconsin</i> | IS | - | 0.7063 | 0.0000 | 0.0000 | - | 3.0000 | 2.0000 | 2.7332 | - | 0.5466 | 0.0000 | 0.0000 | | |
| | RIS | - | - | 0.0000 | 0.0000 | - | - | 2.4599 | 3.5750 | - | - | 0.0000 | 0.0000 | | |
| | LEX | - | - | - | 0.5338 | - | - | 3.2565 | - | - | - | 0.0000 | 0.0000 | | |
| | KW-IS | - | - | - | - | - | - | - | - | - | - | - | 0.5765 | | |
| | GP-STD | 0.0000 | 0.0000 | 1.1860 | 0.0558 | 0.0021 | 0.0244 | 1.1860 | 1.3956 | - | 0.0000 | 0.0001 | 0.2716 | 0.2883 | |
| <i>Parkinson's</i> | IS | - | 1.1860 | 0.0000 | 0.0003 | - | 1.3956 | 0.0001 | 0.0003 | - | 0.7150 | 0.0000 | 0.0000 | | |
| | RIS | - | - | 0.0000 | 0.0006 | - | - | 0.0244 | 0.0408 | - | - | 0.0000 | 0.0000 | | |
| | LEX | - | - | - | 0.0494 | - | - | - | 0.3765 | - | - | - | 0.2716 | | |
| | KW-IS | - | - | - | - | - | - | - | - | - | - | - | - | | |
| | GP-STD | 0.0001 | 0.0004 | 1.0000 | 0.2037 | 0.0000 | 0.0005 | 1.0000 | 0.2716 | - | 0.0001 | 0.0000 | 0.5466 | 0.0030 | |
| <i>Pima Indians Diabetes</i> | IS | - | 0.2037 | 0.0003 | 0.0096 | - | 0.2477 | 0.0000 | 0.0005 | - | 0.5466 | 0.0000 | 0.0013 | | |
| | RIS | - | - | 0.0000 | 0.1643 | - | 0.0661 | 0.2840 | - | - | 0.0000 | 0.0004 | - | | |
| | LEX | - | - | - | 0.0789 | - | - | 0.3466 | - | - | - | 0.0013 | - | | |
| | KW-IS | - | - | - | - | - | - | - | - | - | - | - | - | | |
| | GP-STD | 2.7425 | 2.0493 | 1.6830 | 1.4536 | 0.0244 | 0.0000 | 0.0000 | 0.0001 | - | 0.0061 | 0.0000 | 0.0000 | | |
| <i>Indian Liver Patient</i> | IS | - | 3.1897 | 2.2548 | 1.4536 | - | 0.0529 | 0.1358 | 0.0529 | - | 0.0018 | 0.0061 | 0.0423 | | |
| | RIS | - | - | 3.7213 | 1.0829 | - | 0.0244 | 0.0473 | - | - | 0.1358 | 0.4652 | - | | |
| | LEX | - | - | - | 3.7213 | - | - | 0.2733 | - | - | - | - | 0.0423 | | |
| | KW-IS | - | - | - | - | - | - | - | - | - | - | - | - | | |
| | GP-STD | 0.9304 | 0.0005 | 0.8200 | 0.3787 | 3.2565 | 0.6789 | 2.1866 | 0.6789 | - | 0.0212 | 0.0000 | 0.0002 | 0.0000 | |
| <i>Retinopathy</i> | IS | - | 0.0529 | 0.9304 | 0.0374 | - | 1.4300 | 3.5750 | 3.2565 | - | 0.0000 | 0.0000 | 0.0000 | | |
| | RIS | - | - | 0.0001 | 0.0000 | - | - | 3.5750 | 2.8600 | - | - | 0.0000 | 0.0212 | | |
| | LEX | - | - | - | 0.0374 | - | - | 2.1450 | - | - | - | 0.0000 | - | | |
| | KW-IS | - | - | - | - | - | - | - | - | - | - | - | - | | |
| | GP-STD | 0.8520 | 0.8520 | 1.3555 | 0.4109 | 3.5750 | 0.1059 | 3.0000 | 1.1530 | - | 0.2883 | 0.0001 | 0.0174 | 0.0005 | |
| <i>Vertebral Column 2C</i> | IS | - | 2.2485 | 1.4300 | 1.9039 | - | 0.6110 | 1.9733 | 3.5750 | - | 0.0318 | 0.2883 | 0.0000 | | |
| | RIS | - | - | 2.3099 | 2.1164 | - | 2.7913 | 3.0000 | - | - | - | 0.0174 | 0.0061 | | |
| | LEX | - | - | - | 2.3099 | - | - | 2.0000 | - | - | - | - | 0.0005 | | |
| | KW-IS | - | - | - | - | - | - | - | - | - | - | - | - | | |
| | GP-STD | - | - | - | - | - | - | - | - | - | - | - | - | | |

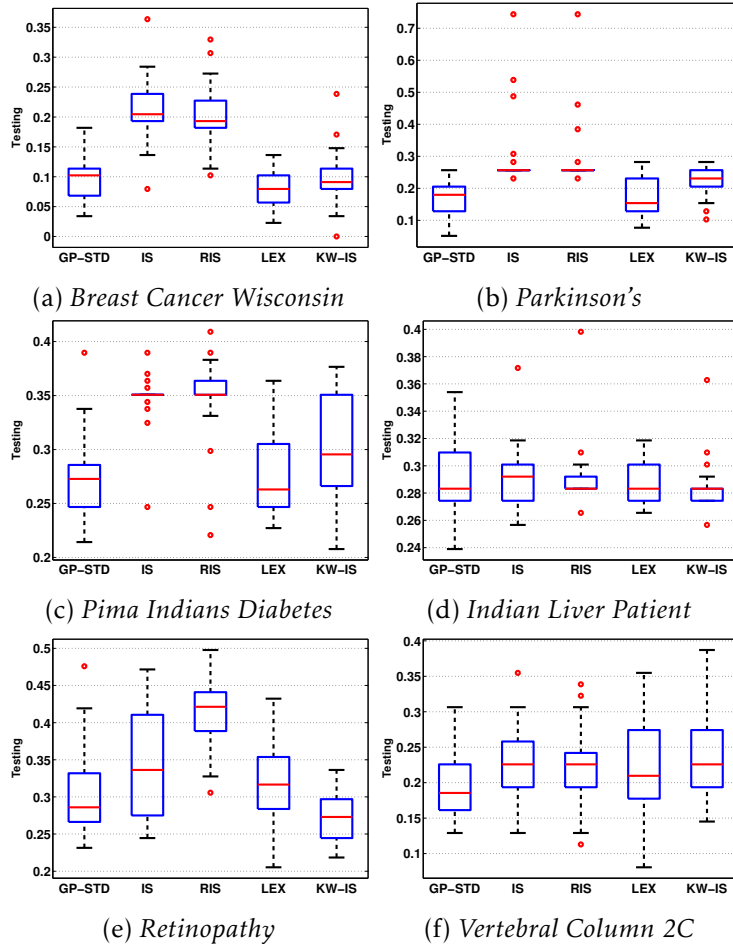


Figure 2.17: Box plot comparison about the test performance of the methods, from the best solution found for each real-world classification problem over all thirty run.

DISCUSSION This experimental work presents the first extensive comparative study between fitness-case sampling methods for GP, that use only a subset of training instances in each generation to reduce computational cost and possibly improve generalization or reduce bloat. In particular, four methods are evaluated Interleaved Sampling (IS), Ran-

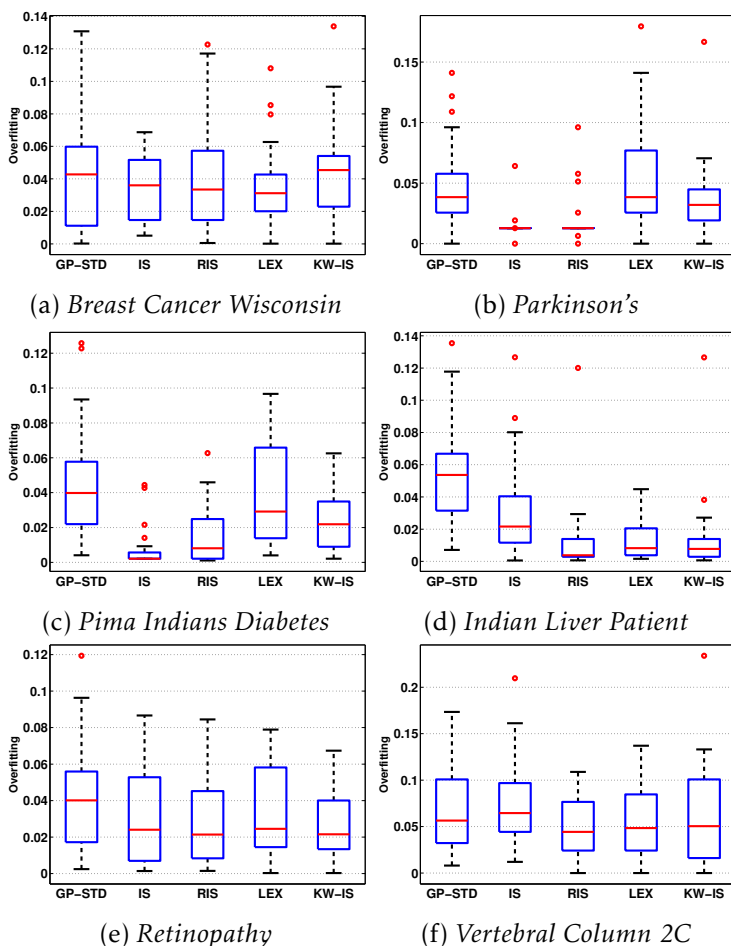


Figure 2.18: Box plot comparison about the overfitting performance of the methods, from the best solution found for each real-world classification problem over all thirty run.

dom Interleaved Sampling (RIS), Lexicase Selection (LEX) and Keep-Worst IS (KW-IS), all of them compared with a standard GP search.

Experimental work is extensive, considering symbolic regression with 5 synthetic problems and 6 real-world problems, as well as supervised classification with 5 synthetic problems and 6 real-world datasets.

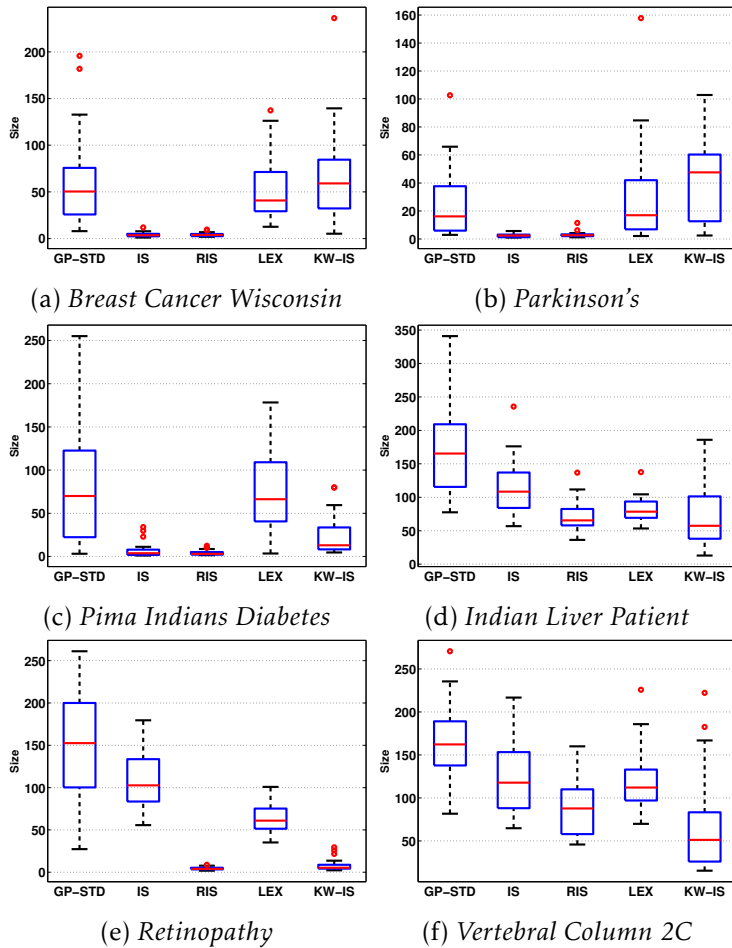


Figure 2.19: Box plot comparison about the average size performance of the methods, from the solutions found for each real-world classification problem over all thirty run.

The algorithms were compared using three performance measures: test error, overfitting and average program size. Statistical comparisons were carried out using a non-parametric multigroup test, and a post hoc non-parametric pairwise testing, in both cases at the 95% confidence level.

The results are illustrative and can be summarized as follows. For symbolic regression the conclusions are dependent on the type of problem. For the simpler benchmark problems, none of the sampling methods outperform standard GP, and only LEX achieves equal performance on all problems. However, when we increase difficulty and consider the real-world problems, then we see the added benefit of the sampling approaches, with three of the methods (LEX, KW-IS and IS) significantly improving upon the standard GP approach. The sampling techniques also exhibit substantially smaller amounts of overfitting and also tend to produce smaller trees than standard GP. Based on these results, it is clear that for difficult real-world symbolic regression fitness-case sampling can help improve performance, reduce overfitting and reduce code growth. Furthermore, based on the presented experimental work, the best methods to use are LEX and KW-IS, with IS also exhibiting strong results, and RIS clearly showing the worst performance.

On the other hand, when we consider classification problems, the results are not convincing in favor of the fitness-case sampling methods. In all problems, either synthetic or real-world, none of the tested algorithms could improve upon the performance of standard GP. In fact only two methods achieved the same performance, with LEX and KW-IS never producing worse results. Moreover, while IS and RIS exhibited the smallest amount of overfitting, this result was not satisfactory since their test performance was significantly worse than GP, LEX and KW-IS on most problems. Finally, an unexpected result was that the fitness-case sampling methods showed the same amount of code growth than standard GP, except for IS and RIS, an improvement that is not desirable due to their low test performance.

In conclusion, the main recommendations that can be drawn from these results are the following. First, LEX and KW-IS are useful fitness-case sampling methods that can improve GP performance on difficult real-world symbolic regression problems. Second, while IS and RIS tend to reduce overfitting and bloat, this comes at the cost of worse test performance, therefore they are not recommended for real-world use. Third, for classification tasks standard GP is still recommended, LEX and KW-IS will not degrade or improve performance in this do-

main. Therefore, a real-world GP-based tool should probably include as a configurable option either LEX, KW-IS or both.

3

CLASSIFICATION WITH GP

In machine learning one of the most common tasks is supervised classification (Kotsiantis et al., 2006). The general task can be stated as follows. Given a pattern $v \in \mathbb{R}^P$, assign the correct class label among C distinct classes $\omega_1, \dots, \omega_C$, using a training set \mathbb{T} of P -dimensional patterns with a known label. The idea is to build a mapping $g(v) : \mathbb{R}^P \rightarrow C$, that assigns each pattern v to a corresponding class ω_i , where g is derived based on the evidence provided by \mathbb{T} . GP has been widely used to address this problem (Muñoz et al., 2015; Sotelo et al., 2013; Trujillo et al., 2011a; Z-Flores et al., 2015; Zhang and Smart, 2004, 2006). In general, GP can be applied to classification following three general approaches:

1. Feature selection and construction (Muharram and Smith, 2005; Muñoz et al., 2015; Sherrah et al., 1997; Trujillo et al., 2011a; Zhang and Smart, 2006).
2. Model extraction (Bentley, 2000; Tanigawa and Zhao, 2000; Tsakonas, 2006; Z-Flores et al., 2015; Zhang and Smart, 2004).
3. Learning ensemble classifiers (Hengpraprom and Chongstitvatana, 2008; Imamura et al., 2003; Langdon and Poli, 2002).

Feature selection and construction is also known as preprocessing of the problem data. These approaches use GP to either select the most interesting problem features or to construct new features that simplify the classification problem. These techniques are often described as either filter (Guo et al., 2005; Muharram and Smith, 2005; Trujillo et al.,

2011a; Zhang and Smart, 2006) or wrapper approaches (Muñoz et al., 2015; Sherrah et al., 1997; Smith and Bull, 2005). In the former, feature construction is done independently of the model used to build the classifier, while in the latter fitness assignment is based on the performance of a classifier. On the other hand, model extraction with GP is used to build specific types of classifiers, such as decision trees (Tanigawa and Zhao, 2000; Tsakonas, 2006), classification rules (Bentley, 2000; Qing-Shan et al., 2007) and discriminant functions (Zhang and Smart, 2004). Finally, ensemble classifiers are used to improve the quality of the classification task by using not only a single classifier, but a group of them, each one providing a different output (Hengpraprom and Chongstitvatana, 2008; Imamura et al., 2003; Langdon and Poli, 2002).

3.1 REAL-WORLD APPLICATIONS OF GP CLASSIFIERS

In order to improve understanding of the relevance of the GP classifiers, we present a experimental work about two different approaches, feature construction and model extraction. These experimental work has been over the most interesting problems, real-world scenarios. The goal is to detect the three main stages of an epileptic seizure (Pre-Ictal, Ictal and Post-Ictal) given a short segment of a ECoG signal. This is posed as a classification problem, where the signal segment represents a pattern $\kappa \in \mathbb{R}^n$, with n the total number of sample points which is dependent on the sampling rate and the signal duration. For instance, since the sampling rate during recording is 256 Hz, if we take a 2 second signal then $n = 512$. Then, this can be defined as a supervised learning problem where a training set \mathbb{T} of n -dimensional patterns with a known classification are used to derive a mapping function $g(v) : \mathbb{R}^P \rightarrow C$, where C are the three distinct epilepsy stages.

The proposal is to solve this problem using GP to derive the mapping function g . GP can be used in different ways to solve a supervised classification tasks such as the one presented here, see for instance (Koza, 1994; Eggermont et al., 2004). However, in this work we test

two GP classifiers that have achieved good results in difficult real-world problems, proposed by Zhang and Smart (2006).

3.1.1 *Static Range Selection GP Classifier (SRS-GPC)*

The first approach is called the Static Range Selection GP Classifier or SRS-GPC a model extraction approach. In this approach, \mathbb{R} is divided into C non-overlapping regions, one for each class. Then, GP evolves a mapping $g(v) : \mathbb{R}^P \rightarrow \mathbb{R}$, such that the region in \mathbb{R} where pattern v is mapped to, determines the class to which it belongs. The fitness function is simple, it consists on maximizing the total classification accuracy of g . For the present problem, since there are three classes (the three seizure stages), \mathbb{R} is divided into the following three ranges for each stage: Pre-Ictal $(-\text{inf}, -1]$, Ictal $[-1, 1]$ and Post-Ictal $[1, \text{inf})$. This is a very simple and straightforward GP implementation, that is easy to setup and use. However, an obvious shortcoming is that it requires an a priori definition of the order and size of the region boundaries.

3.1.2 *Probabilistic GP Classifier (PGPC)*

We refer to the second approach as the Probabilistic GP Classifier, or PGPC (Trujillo et al., 2011a; Zhang and Smart, 2006) a feature construction approach. In PGPC, it is assumed that the behavior of h can be modeled using multiple Gaussian distributions, each corresponding to a single class (Zhang and Smart, 2006). The distribution of each class $\mathcal{N}(\mu, \sigma)$ is derived from the examples provided for it in set Υ , by computing the mean μ and standard deviation σ of the outputs obtained from h on these patterns. Then, from the distribution \mathcal{N} of each class a fitness measure can be derived using Fisher's linear discriminant; for a two class problem it proceeds as follows. After the Gaussian distribution \mathcal{N} for each class are derived, a distance is required. In Zhang and

Smart (2006), Zhang and Smart propose a distance measure between both classes as

$$d = 2 * \frac{|\mu_1 - \mu_2|}{\sigma_1 + \sigma_2}, \quad (3.3)$$

where μ_1 and μ_2 are the means of the Gaussian distribution of each class, and σ_1 and σ_2 their standard deviations. When this measure tends to 0, it is the worst case scenario because the mapping of both classes overlap completely, and when it tends to ∞ , it represents the optimal case with maximum separation. In order to rescale the above measure, the fitness for an individual mapping h is given by

$$f_d = \frac{1}{1 + d}. \quad (3.4)$$

After executing the GP, the best individual found determines the parameters for the Gaussian distribution \mathcal{N}_i associated to each class. Then, a new test pattern v is assigned to class i when \mathcal{N}_i gives the maximum probability.

In summary, we use two different GP classifiers, SRS-GPC and PGPC. Both are trained using the epilepsy signal recorded during a single day, from a total of five different days, and then tested on the remaining days. The signal from each day is divided into segments of equal duration, here we build two different partitions, the first with 1 second signals and the other using 2 second signals. The next section presents the experimental setup and main results.

3.1.3 *Experiments and Results*

The goal of the experimental work is to evaluate the accuracy of intra-subject classification of epilepsy signals. In other words, to test the performance of classifiers that are trained and tested with signal samples from a single test subject.

DATA SETS Epileptic signals were recorded from four different test subjects (five rodents) on which the seizures are elicited and the signals recorded; call them subjects S_1 , S_2 , S_3 and S_4 . For each subject,

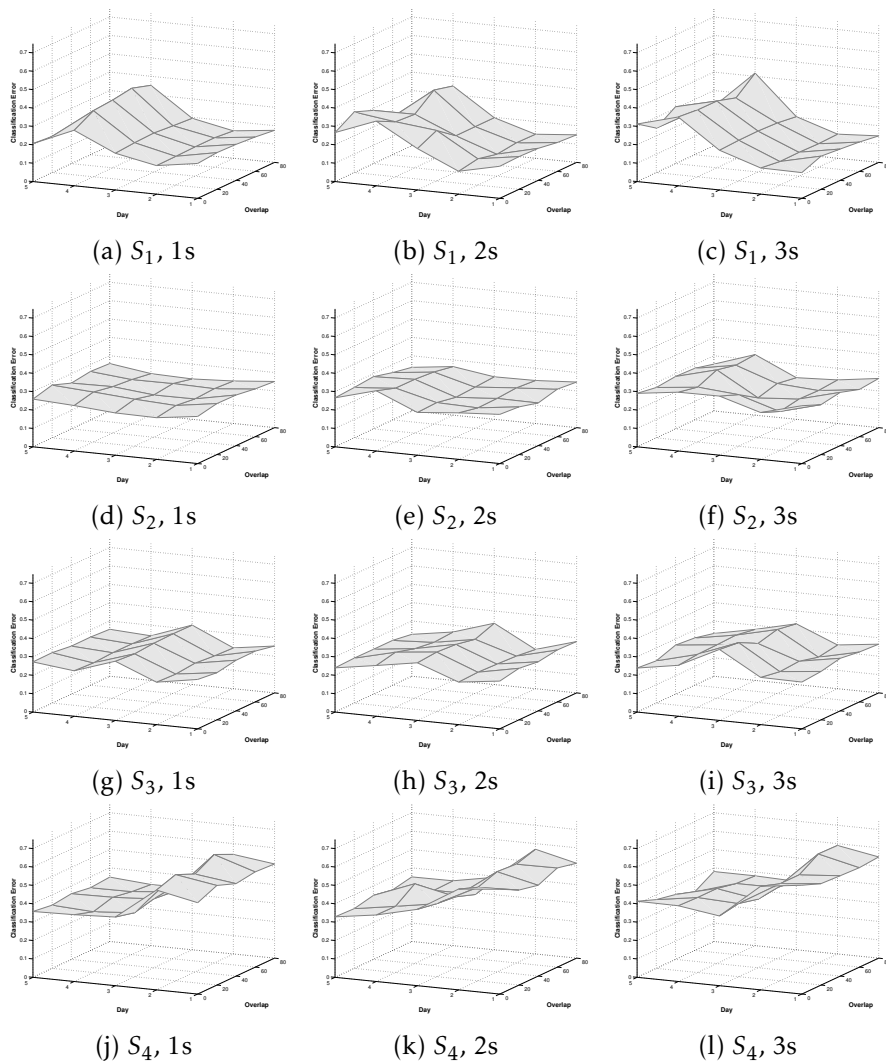


Figure 3.1: Median classification error plotted against day used for training and amount of overlap for the SRS-GPC. Each row corresponds to each subject (top row is S_1 and bottom row S_4) and each column corresponds to a different segment duration; leftmost row is 1 sec., middle row 2 and rightmost 3.

CLASSIFICATION WITH GP

Table 3.1: Parameters for the PGPC system used in the experimental tests.

| Parameter | Description |
|-------------------------------|--|
| <i>Population size</i> | 200 individuals. |
| <i>Generations</i> | 200 generations. |
| <i>Initialization</i> | <i>Ramped Half-and-Half</i> , with 6 levels of maximum depth. |
| <i>Operator probabilities</i> | Crossover $p_c = 0.8$; Mutation $p_\mu = 0.2$. |
| <i>Function set</i> | $\{+, -, *, /, \sqrt{\cdot}, \sin, \cos, \log, x^y, \cdot , if\}$ |
| <i>Terminal set</i> | $v_\mu, v_m, v_\sigma, v_{max}, v_{min}, v_s$ and v_k |
| <i>Bloat control</i> | Dynamic depth control. |
| <i>Initial dynamic depth</i> | 6 levels. |
| <i>Hard maximum depth</i> | 20 levels. |
| <i>Selection</i> | Lexicographic parsimony tournament |
| <i>Survival</i> | Keep best elitism |

a level-5 seizure is elicited and recorded on five consecutive days; call them *Day-1*, *Day-2*, *Day-3*, *Day-4* and *Day-5*. Afterwards, the signal is classified manually by a human expert, who specifies where each epilepsy stage begins and ends. This manual classification establishes the ground-truth for the supervised-learning problem. The signal is divided into C segments, each constituting a sample from the corresponding stage. We test three different segment durations: 1, 2 and 3 seconds. When the signal is divided, we allow for different amounts of overlap between consecutive segments given by a percentage of signal duration. Five different overlaps are tested: 0%, 20%, 40%, 60% and 80%. It is important to state that signal segments that lie on two adjacent stages are removed from the data-sets.

GP IMPLEMENTATION Both GP classifiers use a standard Koza style tree based representation, with subtree-crossover and sub-tree mutation. The basic parameters of both systems is presented in Table 3.1.

For both GP classifiers, the terminal elements are basic statistical features computed for each signal segment x that is to be classified. Specifically, the terminal set contains: mean value v_μ , median v_m , standard deviation v_σ , maximum v_{max} , minimum v_{min} , skewness v_s and kurtosis v_k . For each experimental configuration, 30 independent runs of the GP system are executed. This is necessary, since GP is a non-deterministic search process.

PERFORMANCE MEASURES To gain a deeper understanding of the GP-based classifiers, a detailed analysis of the results is presented using three standard performance measures computed on the test set of data: classification error, sensitivity and specificity. The first measure is explicit, while the latter two are derived from the confusion matrix (true positives (TP), true negatives (TN), false positives (FP), false negatives (FN)) generated by the classifier with respect to each class; given by:

$$\text{Sensitivity} : S = \frac{TP}{TP + FP} . \quad (3.5)$$

$$\text{Specificity} : Sp = \frac{TN}{FN + TN} . \quad (3.6)$$

INTRA-SUBJECT CLASSIFICATION As stated above, the task of automatic epileptic stage identification is posed a supervised learning problem. The training data used consists of all of the signal segments taken from a single recording day. Then, for intra-subject classification the evolved classifiers are tested on the signal segments from each additional recording day. This is done for all subjects (S_1 , S_2 , S_3 and S_4) and all possible combinations of training and testing days, as well as for all segment durations (1,2 or 3 seconds) and overlaps (0, 20, 40, 60 and 80%).

Figure 3.1 shows the median classification error achieved by the SRS-GPC. The error is plotted with respect to the day used to train the classifier and the amount of overlap. Each row corresponds to each subject (top row is S_1 and so on) and each column corresponds to a

CLASSIFICATION WITH GP

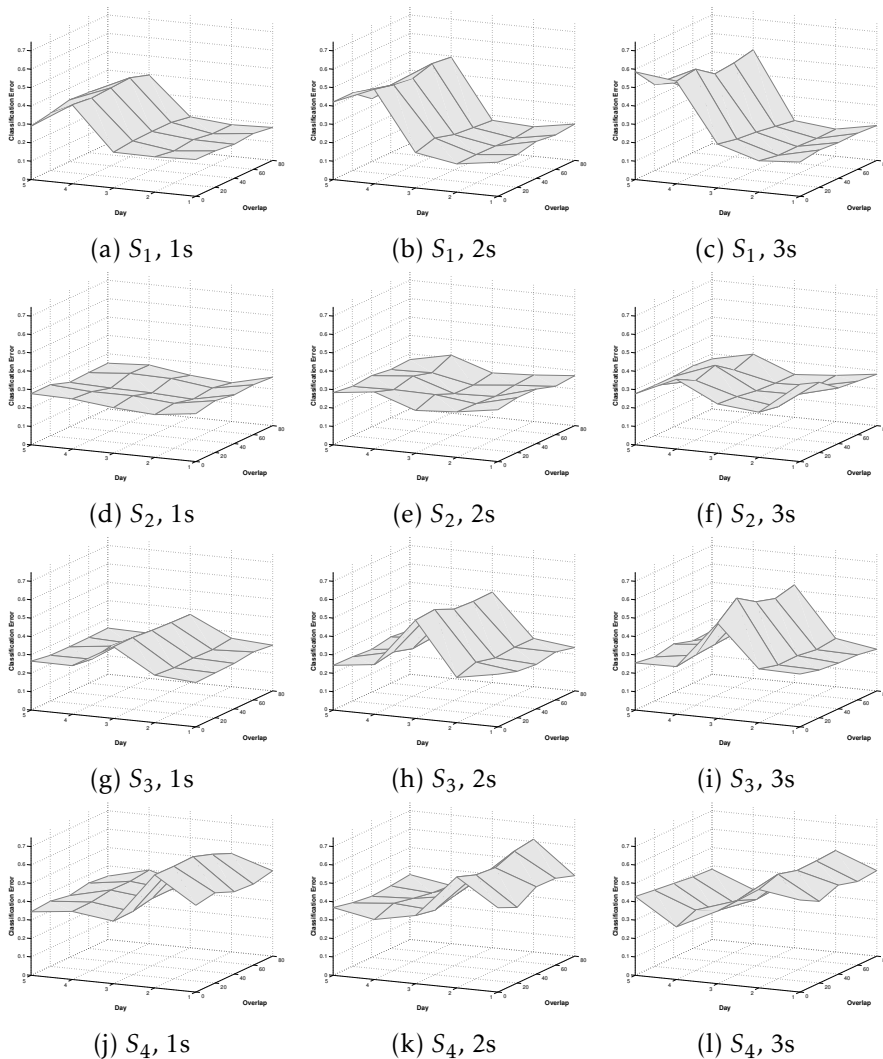


Figure 3.2: Median classification error plotted against day used for training and amount of overlap for the PGPC. Each row corresponds to each subject (top row is S_1 and bottom row S_4) and each column corresponds to a different segment duration; leftmost row is 1 sec., middle row 2 and rightmost 3.

different segment duration (leftmost row is 1 sec. and so on). Figure 3.2 presents the same for PGPC.

Figures 3.3 and 3.4 present the results for the median sensitivity and specificity respectively, for SRS-GPC. Similarly, figures 3.5 and 3.6 show the same results for PGPC.

DISCUSSION The plots shown in Figures 3.1 - 3.6 present an extensive empirical evaluation of the proposed approach for stage identification. The best way to read these plots is as follows. In all cases, the flatness of the plotted surfaces is related to the robustness of the proposed approach to the different data configurations. Moreover, for classification error lower values are best, and for sensitivity and specificity the converse is true.

The plots shown above illustrate strong statistical trends regarding the behavior of the system, from which several noticeable implications can be derived. First, overall the best performance is achieved by the simpler SRS-GPC classifier, compared with PGPC. This was slightly unexpected, given the strong assumptions made by the SRS-GPC method. Nonetheless, across all configurations and all test subjects, SRS-GPC is better based on all three performance measures. Second, it appears that the segment length is an important determining factor in classification performance. In particular, smaller segments (1 or 2 seconds) are easier to classify than longer ones (3 s). Also, it appears that a higher amount of overlap between segments can induce better results. A reasonable explanation for both observations can be given based on the machine learning approach followed by the proposal. In particular, shorter segments and a larger overlap produce more training data with which to train the classifier and thus obtain a better supervised learning process.

Third, while the results described above hold for all test subjects, there are still some subtle and important differences among them. The GP classifiers achieve good results for three of the test subjects (S_1 , S_2 and S_3), evidenced by all three performance measures. However, the performance is considerably worse for test subject S_4 . This result is coherent with the general observation that different subjects can produce quite different signal patterns, even if all signals appear similar at a

CLASSIFICATION WITH GP

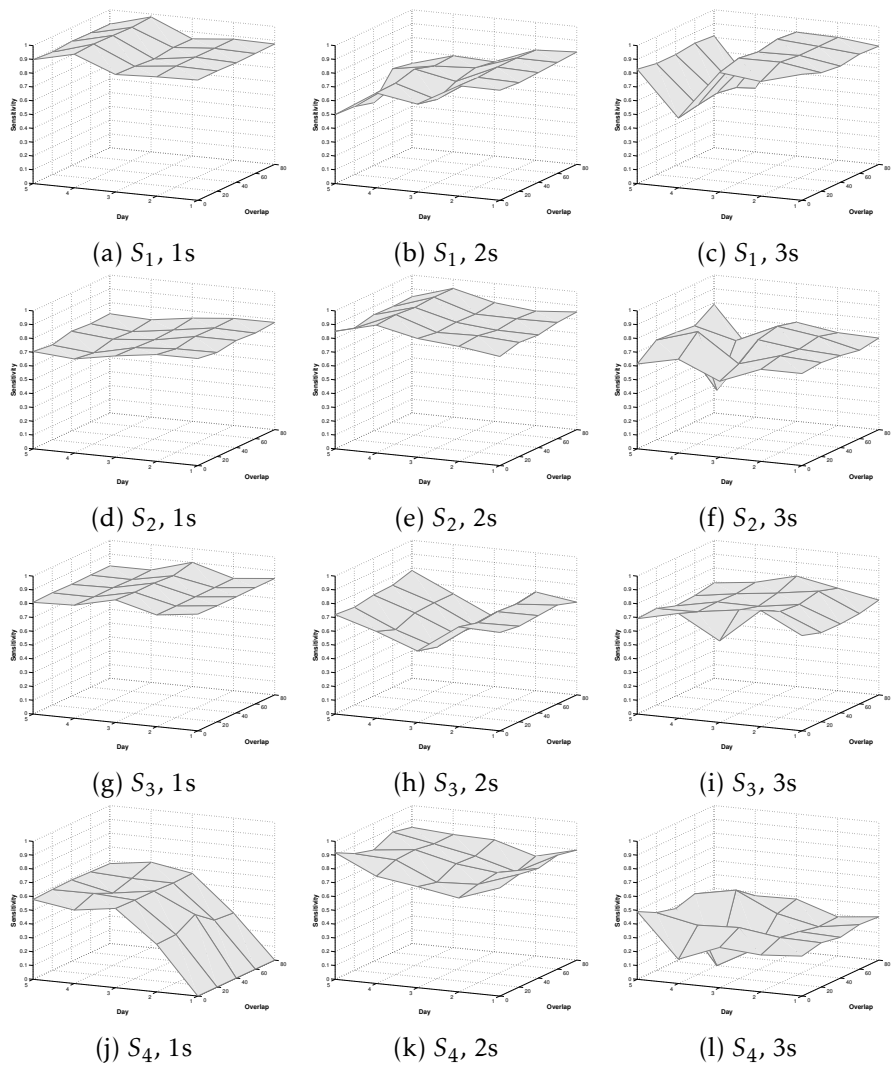


Figure 3.3: Median sensitivity plotted against the day-used for training and the amount of overlap for the SRS-GPC. Each row corresponds to each subject (top row is S_1 and bottom row S_4) and each column corresponds to a different segment duration; leftmost row is 1 sec., middle row 2 and rightmost 3.

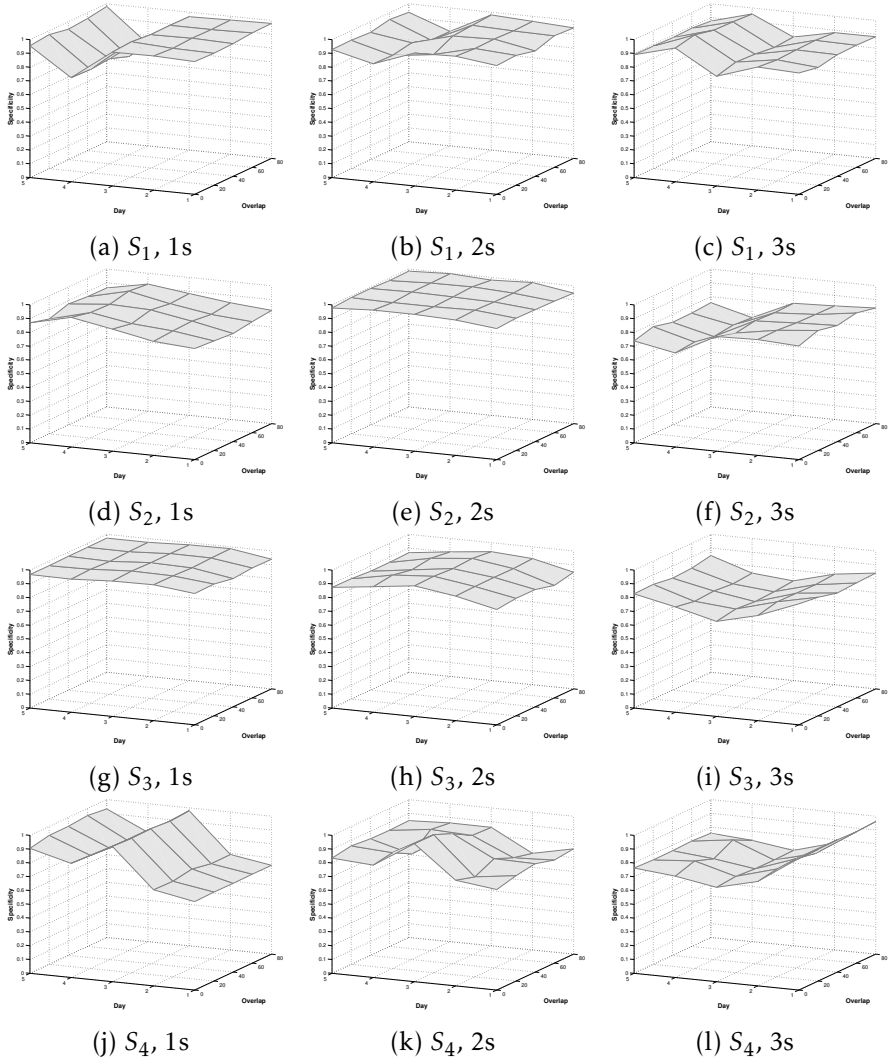


Figure 3.4: Median specificity plotted against the day used for training and the amount of overlap for the SRS-GPC. Each row corresponds to each subject (top row is S_1 and bottom row S_4) and each column corresponds to a different segment duration; leftmost row is 1 sec., middle row 2 and rightmost 3.

CLASSIFICATION WITH GP

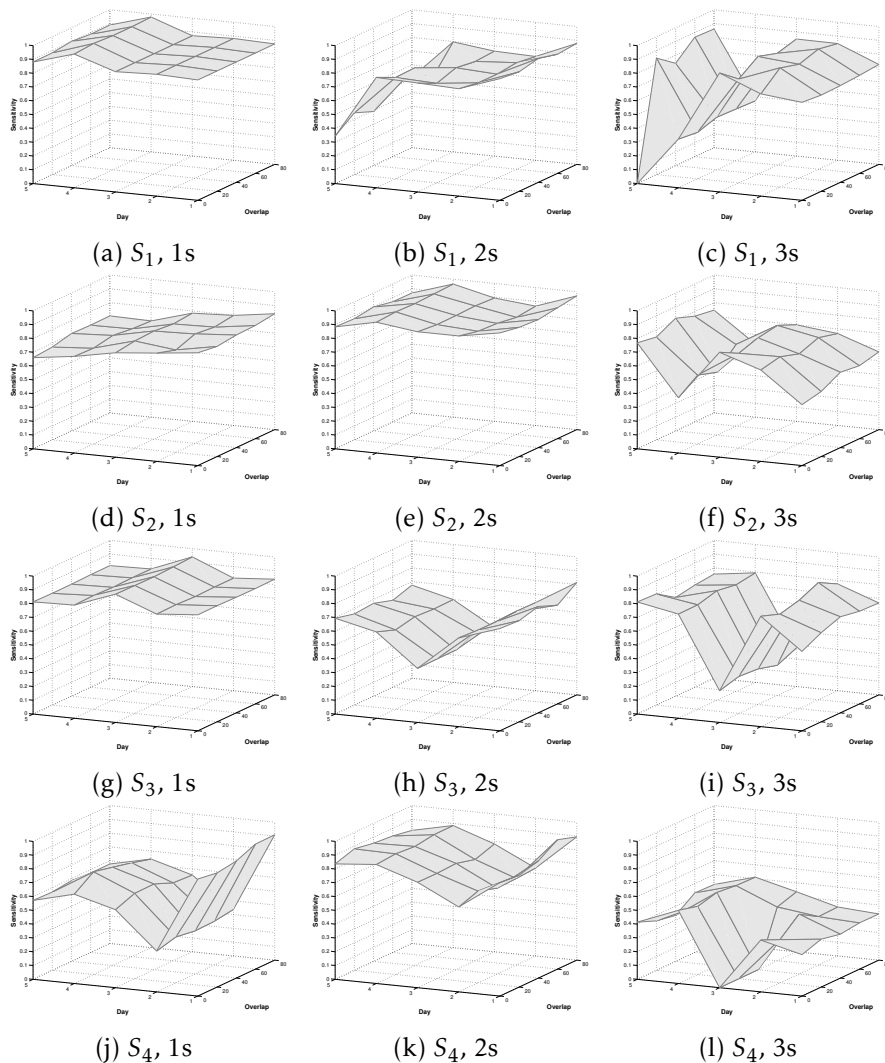


Figure 3.5: Median sensitivity plotted against the day used for training and the amount of overlap for the PGPC. Each row corresponds to each subject (top row is S_1 and bottom row S_4) and each column corresponds to a different segment duration; leftmost row is 1 sec., middle row 2 and rightmost 3.

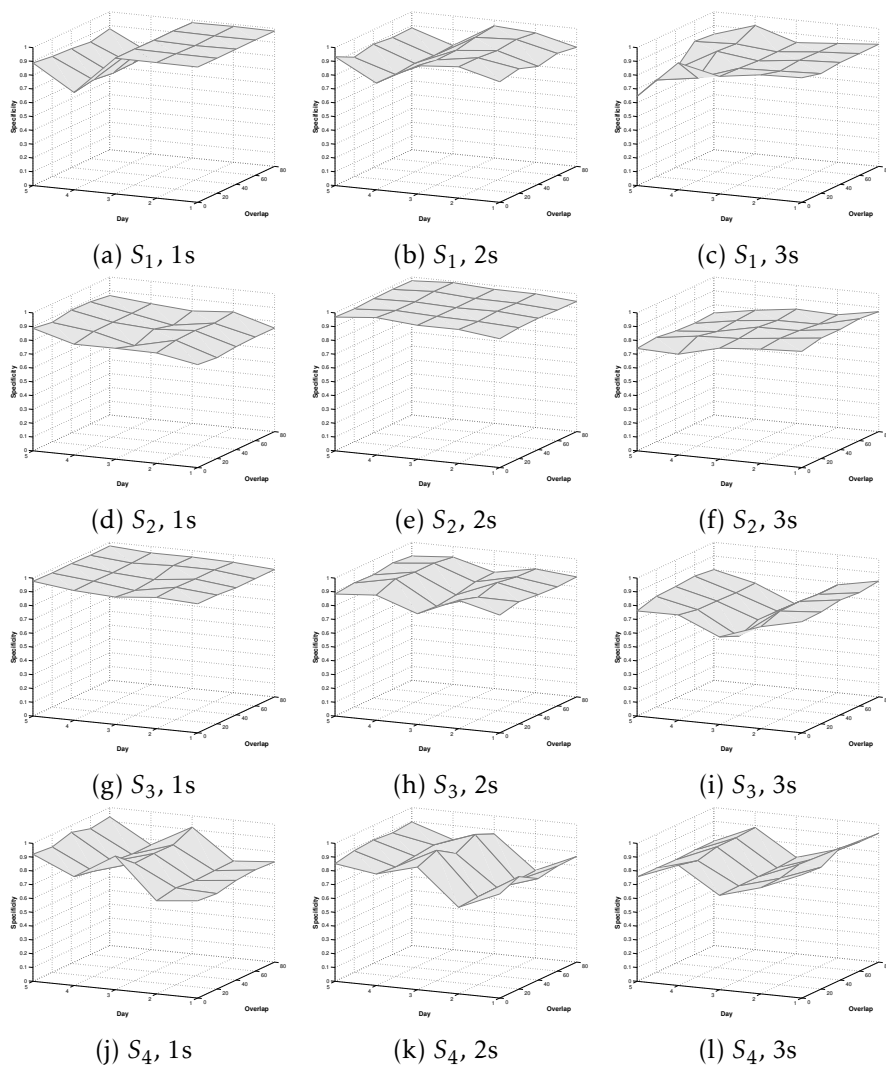


Figure 3.6: Median specificity plotted against the day used for training and the amount of overlap for the PGPC. Each row corresponds to each subject (top row is S_1 and bottom row S_4) and each column corresponds to a different segment duration; leftmost row is 1 sec., middle row 2 and rightmost 3.

coarser scale. Moreover, the plots show that classifier performance can sometimes vary depending upon which training day is used. In effect, this shows that, of course, the recorded signals should not be expected to be homogenous given the different (even if slight) conditions under which each recording session is done.

Finally, considering results obtained in previous work using GADs (Sotelo et al., 2007, 2008), here we find that the ability to identify epilepsy stages are substantively better with the GP classifiers. The results suggest that the time domain properties used in GP were more efficient for stage classification than the signal complexity computed from the time-frequency domain. Moreover, such features, and the evolved classifiers, are of low computational complexity and could be easily implemented on dedicated hardware.

Here is presented an automatic method for identifying the three main stages of an epileptic seizure from ECoG signals. The proposal is based on posing the problem as a supervised learning problem and solving it with Genetic Programming. The results exhibit strong statistical tendencies of the GP classifiers that suggest that the approach is able to solve the intra-patient classification problem. These results are unique and show substantial improvement when compared with previous methods (Sotelo et al., 2007, 2008, 2012).

4

THE ESTIMATION OF PROBLEM DIFFICULTY: RELATED WORK

Determining problem difficulty has been an important issue in EC for several years (McClymont et al., 2012). From an algorithmic perspective, problem difficulty can be related to the total runtime (or memory) required to find an optimal solution. Recently, He et al. (2015) took this view one step further, to analytically define broad classes of fitness functions which allowed them to demonstrate that easy functions define unimodal fitness landscapes, while hard functions define deceptive landscapes for a (1+1) ES. However, it is important to remember that the difficulty of a particular problem depends upon the solution method. Therefore, in what follows we will try to limit our overview to GP-related research.

4.1 EVOLVABILITY INDICATORS

The fitness landscape has dominated the way geneticists think about biological evolution and has been adopted by the EC community as a way to visualize evolution dynamics (Wright, 1932). Formally, a fitness landscape can be defined as a triplet (ω, χ, f) , where ω is a set of configurations, χ is a notion of neighborhood, distance or accessibility on ω , and f is a fitness function (Stadler, 2002). The local and global structure of the fitness landscape describes the underlying difficulty of a search. However, in the case of standard GP (Langdon and Poli, 2002) the concept of a fitness landscape is not clearly defined (Kinnear, 1994). To overcome this, some works have constructed synthetic

THE ESTIMATION OF PROBLEM DIFFICULTY: RELATED WORK

problems; such as the Royal Tree problem (Punch et al., 1996) or the K-landscapes model (Vanneschi et al., 2011), where the goal of the search is defined as a particular tree structure with a specific syntax. Unfortunately, such models are not realistic since the space of possible programs is highly redundant (Langdon and Poli, 2002) in most domains, and the goal is not a particular syntax but a particular expected output, also known as semantics (McPhee et al., 2008; Vanneschi et al., 2014). Therefore, some researchers have proposed variants of GP that explicitly account for program semantics. In semantic space the fitness landscape is clearly defined and unimodal. This has lead researchers to develop specialized search operators that modify program syntax while geometrically bounding the semantics of the generated offspring, this is known as geometric semantic GP (GSGP) (Moraglio et al., 2012). Nevertheless, such approaches are still problematic since the size of the evolved programs grows exponentially with every generation, a limitation that is not easily solved (Silva and Costa, 2009). This work will focus on measures of problem difficulty for standard GP systems (Koza, 1992), but could be applied to other supervised learning systems including GSGP.

In general, most meta-heuristics work under the assumption that the fitness of a candidate solution, a point on the fitness landscape, is positively correlated with the fitness of its (some) neighbors. Such a property can be defined as the *evolvability* of a landscape (Altenberg, 1994; O'Neill et al., 2010). EIs extract a numerical indicator of a specific property of the fitness landscape to provide a measure of the evolvability within the landscape. Malan and Engelbrecht (2013) presents a comprehensive survey of EIs and other forms of fitness landscape analysis. Those that have been studied in GP literature include neutrality (Galván-López et al., 2008; Kimura, 1983), locality (Galván-López et al., 2010; Rothlauf, 2006), ruggedness (Kauffman and Levin, 1987; Vanneschi et al., 2011), fitness distance correlation (FDC) (Clergue et al., 2002; Jones and Forrest, 1995; Tomassini et al., 2005), fitness clouds (Verel et al., 2003) and the negative slope coefficient (NSC) (Vanneschi et al., 2004). While these approaches can sometimes provide good estimates of problem difficult for GP, they suffer from two practical

limitations. First, for each new problem instance they require a large amount of data, by sampling the search space or performing several runs. Second, they cannot estimate the actual quality of the solution found, which can be important if we want to choose the best algorithm to use for a new problem, and if such a choice must be made in real-time. Indeed, Malan and Engelbrecht (2013) point out that a possible way forward is to build a mapping that can estimate algorithm performance based on a set of descriptive features of the problem, an approach that would provide a more practical measure of problem difficulty and allow us to choose the best algorithm for the specific task. Malan and Engelbrecht (2014) attempted to find a link between EIs and algorithm performance for particle swarm optimization.

4.2 PERFORMANCE PREDICTION

PEPs predict the performance of a GP search on an unseen problem instance without performing the search or sampling the solution space. These models have been derived using a machine learning approach (Graff et al., 2013a; Graff and Poli, 2008; Martínez et al., 2012; Trujillo et al., 2011a,b). The performance of GP on a set of problems and a description of those problems are used to pose a supervised learning task. A promising feature of PEPs is that they are not only useful for GP, they can also be used to predict the performance of other algorithms (Graff and Poli, 2010; Trujillo et al., 2011b).

Graff and Poli (2010) proposed linear predictive models based on a sampling of the fitness landscape, given by

$$\Psi(v) \approx a_0 + \sum_{\xi \in \Xi} a_\xi \cdot d(\xi, v), \quad (4.7)$$

where $\Psi(v)$ is the predicted performance, v is the target functionality, $d(\xi, v)$ is a distance measure¹, Ξ is the set of all possible program outputs, also known as semantic space (Moraglio et al., 2012), and where

¹Such a distance measure is a common fitness function for many application domains of GP, particularly for symbolic regression problems.

THE ESTIMATION OF PROBLEM DIFFICULTY: RELATED WORK

each ξ represents the vector of program outputs obtained from the set of fitness cases used to define a particular problem, also known as the semantics of the program (McPhee et al., 2008). In other words, Graff and Poli (2010) derive PEPs by sampling semantic space Ξ . These models were tested on symbolic regression and 4-input Boolean problems with promising results.

The second and more recent approach towards building a PEP focuses on the problem data (Graff et al., 2013a; Graff and Poli, 2011; Graff et al., 2013b; Martínez et al., 2012; Trujillo et al., 2011a, 2012, 2011b,c) and proceeds as follows. Assume we want to solve a supervised learning problem p with a GP search, where fitness is given by a cost function that must be minimized, such as an error measure. Let us define the performance of the GP algorithm as the associated error of the best solution found during training when it is evaluated on a particular set of fitness cases T , call this quantity $F_T(p)$. The goal is to predict $F_T(p)$, so first we construct a feature vector $\beta = (\beta_1, \beta_2, \dots, \beta_N)$ of N distinct features that describe the main properties of p . Then, a PEP is function K such that

$$F_T(p) \approx K(\beta). \quad (4.8)$$

Notice that the form of K is not a priori restricted in any way. Graff and Poli (2011) use a linear function similar to the one used in their previous work (Graff and Poli, 2010). Using this approach the feature vector β should be designed specifically for the domain of p . For example, features designed for symbolic regression and Boolean problems are proposed in Graff and Poli (2011), and the results show that the predictive accuracy surpasses that of the fitness-based models proposed in Graff and Poli (2010). However, their work did not scale well to real-world cases. For instance, in Graff et al. (2013a,b) the authors built PEPs to predict performance on real-world problems, but require information obtained from runs performed on similar problem instances, models built with simpler synthetic problems could not be used. It was not trivial to map multidimensional problems to the proposed feature space since the training problems were much simpler with a small number of dimensions. It would be impractical to consider all possi-

ble dimensionalities during training. This is an important limitation in building PEPs, since it is not trivial to have all the possible versions of the same problem. Moreover, in the proposals made by Graff et al. (2013a); Graff and Poli (2011); Graff et al. (2013b) the models predicted the performance of the GP system on the training set of fitness cases; i.e., T was the training set. While certainly of importance, performance on the training set may not be useful if the algorithm overfits the training examples, which happens often in real-world scenarios.

In previous work Trujillo et al. (2011a,b,c), we used a similar approach to predict the performance of a GP-classifier using descriptive features that characterize the geometry of the data distribution in feature space. The PEPs were built using quadratic linear models and non-linear GP models, the latter achieving the best performance on synthetic problems. However, it was not clear how well the PEPs generalized to unseen problem instances, particularly to real-world problems with imbalanced datasets and larger feature spaces than those used to train the models, a similar difficulty pointed out in Graff et al. (2013a,b). The current work extends our previous contributions by performing the learning process on 2D synthetic problems and testing on a wide variety of real-world datasets. Moreover, an important contribution of this work is that the PEP models are used to predict the performance of the best solution found by GP when it is evaluated on the test set of data. To achieve improved performance this work also proposes a two-tiered ensemble approach using specialized PEP models and a preprocessing stage for dimensionality reduction.

**THE ESTIMATION OF PROBLEM DIFFICULTY:
RELATED WORK**

5

PEP: PREDICTOR OF EXPECTED PERFORMANCE

The general goal of this work is to build models that can predict the performance of a GP-classifier (PGPC) without executing the search or sampling the problem's search space. The general proposal is depicted in Figure 5.1, where for a given classification problem we do the following. First, apply a preprocessing step to simplify the feature extraction process and deal with multidimensional representations. Second, perform feature extraction to obtain an abstraction of the problem. Third, use a PEP model that takes as input the extracted features and produces as output the predicted classification error (PCE) on the testing set.

Moreover, to derive the PEP models we use a supervised learning methodology, depicted in Figure 5.2. This process takes as input a set of synthetic classification problems \mathcal{Q} and produces as output the PEP model as follows:

1. Compute the average classification error (CE_{μ}) on the test data by PGPC for each $p \in \mathcal{Q}$.
2. Apply a preprocessing for dimensionality reduction using principal component analysis (PCA), and take the first m principal components to represent the problem data.
3. Perform feature extraction on the transformed data using statistical and complexity measures to build a feature vector β for each $p \in \mathcal{Q}$.

PEP: PREDICTOR OF EXPECTED PERFORMANCE

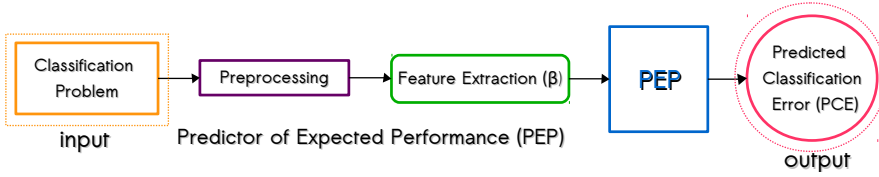


Figure 5.1: Block diagram of the proposed PEP approach. Given a classification problem, the goal is to predict the performance of a GP classifier on the test data, in this case PGPC.

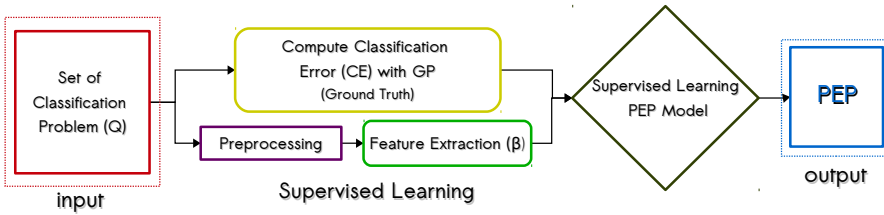


Figure 5.2: The methodology used to build the PEP model. Given a set \mathcal{Q} of synthetic classification problems: (1) compute the CE_{μ} of PGPC on all problems; (2) apply a preprocessing for dimensionality reduction; (3) extract the feature vector β from the problem data; and (4) learn the predictive model using GP.

4. Finally, using the set of feature vector/performance pairs $\{(\beta_i, CE_{\mu_i})\}$, formulate a supervised symbolic regression problem and solve it using GP.

5.1 SYNTHETIC CLASSIFICATION PROBLEMS

A set of synthetic classification problems was generated to learn our PEP models. Specifically, 500 binary classification problems were generated using Gaussian mixture models (GMMs) with either unimodal or multimodal classes, with different amounts of class overlap. All class samples lie within the closed 2-D interval $v, v \in [-10, 10]$, and 200 sample points were randomly generated for each class. The parameters for

the GMM of each class were randomly chosen using a uniform distribution in the following ranges:

1. Number of Gaussian components: $\{1, 2, 3\}$.
2. Median of each Gaussian component for each dimension: $[-3, 3]$.
3. Each element of the covariant matrix of each Gaussian component: $(0, 2]$.
4. The rotation angle of each covariance matrix: $[0, 2\pi]$.
5. Proportion of samples generated with each Gaussian component: $[0, 1]$.

5.2 PGPC CLASSIFICATION ERROR

For each problem $p \in \mathcal{Q}$ we perform 30 runs of PGPC, randomly choosing the training and testing sets in each run. Then, the mean classification error $CE\mu$ is computed by the average of the test performance achieved by the best solutions found in each run. The parameters of the PGPC system are given in Table 5.1, a tree-based GP algorithm with dynamic depth bloat control (Silva and Costa, 2009), implemented using Matlab and the GPLAB toolbox (Silva and Almeida, 2003). Figure 5.3 presents some examples, showing the problem data, the $CE\mu$ achieved by PGPC and the standard deviation σ over all runs. The problems are ordered from the lowest $CE\mu$ (easiest problem, depict in Fig. 5.3(a)) to the highest, $CE\mu$ (hardest problem, depict in Fig. 5.3(f)).

Figure 5.4 summarizes PGPC performance over all 500 synthetic problems in \mathcal{Q} . Figure 5.4(a) plots the $CE\mu$ for each problem, ordered from the lowest to the highest error. On the other hand, Figure 5.4(b) shows an histogram of PGPC performance, quantifying how many problems are solved with a particular $CE\mu$. We arbitrarily set a threshold such that problems in the range $0 \leq CE\mu \leq 0.15$ are considered “easy” and the rest are considered to be “hard”. From this perspective the plot reveals that randomly generated problems produce a

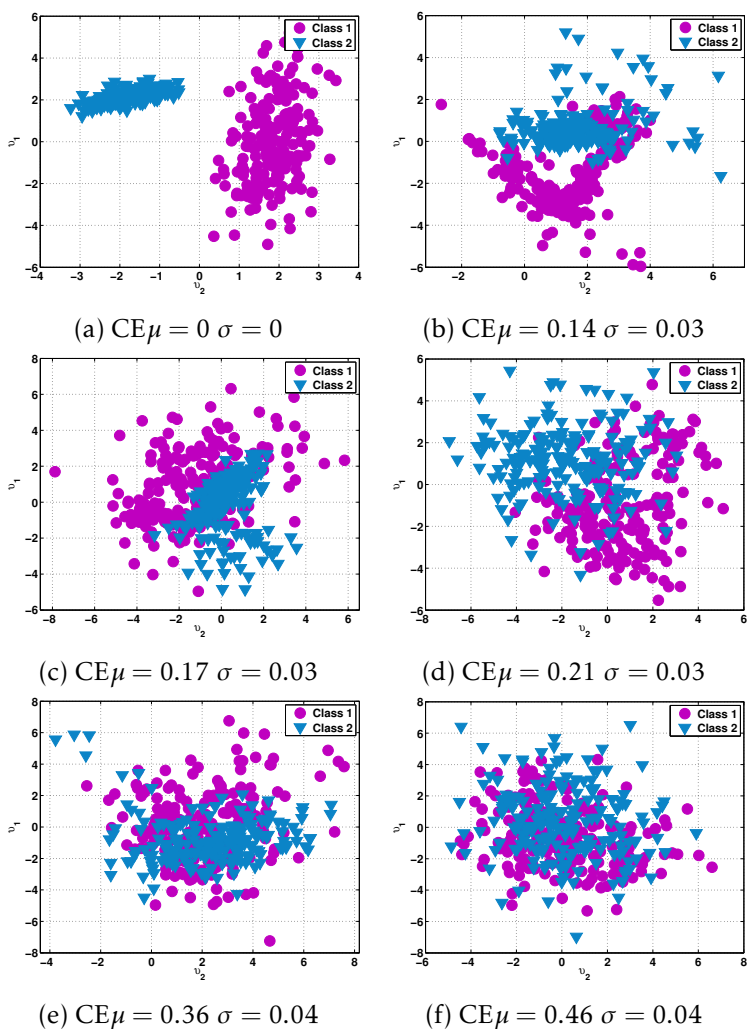


Figure 5.3: The scatter plots show examples of synthetic classification problems, specifying the $CE\mu$ and standard deviation σ achieved by PGPC. These ordered from the lowest $CE\mu$ (easiest depict in Fig. 5.3(a)) to the highest $CE\mu$ (hardest depict in Fig. 5.3(f)).

biased distribution, where most problems are easy to solve. Since we

Table 5.1: Parameters for the PGPC algorithm.

| Parameter | Description |
|-------------------------------|--|
| <i>Population size</i> | 200 individuals |
| <i>Generations</i> | 200 generations |
| <i>Initialization</i> | <i>Ramped half-and-half</i> , with 6 levels of maximum depth |
| <i>Operator probabilities</i> | Crossover $p_c = 0.8$; Mutation $p_\mu = 0.2$ |
| <i>Function set</i> | $\{+, -, *, /, \sqrt{\cdot}, \sin, \cos, \log, x^y, \cdot , if\}$ |
| <i>Terminal set</i> | $\{v_1, \dots, v_i, \dots, v_p\}$ where each v_i is a dimension of the data patterns $v \in \mathbb{R}^P$ |
| <i>Bloat control</i> | Dynamic depth control |
| <i>Initial dynamic depth</i> | 6 levels |
| <i>Hard maximum depth</i> | 20 levels |
| <i>Selection</i> | Tournament, size 3 |
| <i>Survival</i> | Keep best elitism |
| <i>Training Data</i> | 70% |
| <i>Testing Data</i> | 30% |
| <i>Runs</i> | 30 |

intend to use this set to pose a supervised learning task, this would induce an unwanted bias. Therefore, we subsample \mathcal{Q} to get a more balanced distribution over $CE\mu$. The new set consists of 300 problems, and Figure 5.5 summarizes PGPC performance over this new set \mathcal{Q}' . Notice that the performance plot for $\mathcal{Q}' \subset \mathcal{Q}$ is similar to the one obtained for \mathcal{Q} (see Figure 5.5(a)), but now the distribution over $CE\mu$ is flat (Figure 5.5(b)), providing a more balanced learning set.

5.3 PREPROCESSING

Previous work has found that PEP models can predict GP performance accurately for small scale synthetic problems (Graff and Poli, 2008, 2010, 2011; Martínez et al., 2012; Trujillo et al., 2011a, 2012,

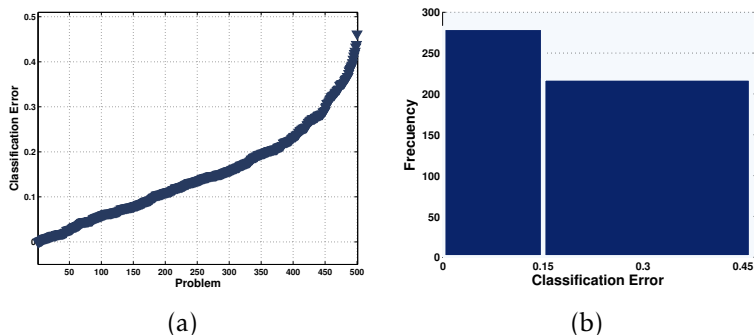


Figure 5.4: Performance of PGPC over all 500 synthetic problems in \mathcal{Q} ; where: (a) shows the $CE\mu$ for each problem, ordered from the easiest to the hardest; and (b) shows the histogram over $CE\mu$.

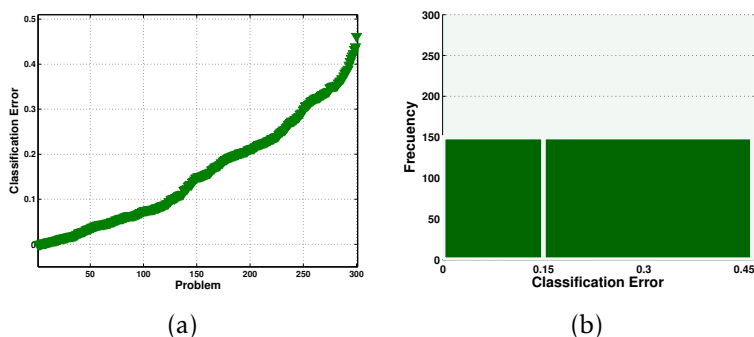


Figure 5.5: Performance of PGPC over all 300 synthetic problems in $\mathcal{Q}' \subset \mathcal{Q}$; where: (a) shows the $CE\mu$ for each problem, ordered from the easiest to the hardest; and (b) shows the histogram over $CE\mu$.

2011b,c), but accuracy degrades for real-world problems with high dimensional data (Graff et al., 2013a,b). This is due to the fact that feature extraction (the next step in the PEP approach) fails at extracting meaningful information in high dimensional spaces (Graff et al., 2013a,b). To deal with this issue, we apply a dimensionality reduction preprocessing of the problem data using PCA (Duda et al., 2000). We propose

to take the first m principal components to represent the data of each problem. In particular, we set $m = 2$ in all experiments reported here. In this way, all problems are reduced to the same number of dimensions used in the synthetic training set.

5.4 FEATURE EXTRACTION

The goal of this step is to extract a set of descriptive measures from each problem. In this work, we use a subset of the features proposed in Sohn (1999) and Ho and Basu (2002). Those works attempted to develop meta-representations of classification problems. A wider set of features was previously tested in Trujillo et al. (2011a, 2012, 2011b,c), but the present work only uses those features that showed the highest correlation with $CE\mu$. We also propose three new descriptors based on the Canberra distance; each measure is presented next.

GEOMETRIC MEAN (SD) measures the homogeneity of covariances (Michie et al., 1994; Sohn, 1999). This quantity is related to a test of the hypothesis that all populations have a common covariance structure; i.e.. to the hypothesis $H_0 : \Sigma_1 = \Sigma_2$, which can be tested via Box's M test statistic (MTS) (Anderson, 1958), that can be re-expressed as

$$SD = \exp \left\{ \frac{MTS}{m \sum_{i=1}^C (n_i - 1)} \right\} \quad (5.9)$$

where C is the number of classes, n_i is the number of the instances for i -th class and m is the number of dimensions. The SD is strictly greater than unity if the covariances differ, and is equal to unity if and only if the MTS is zero.

FEATURE EFFICIENCY (FE) measures the amount by which each feature dimension contributes to the separation of both classes. This measure is computed for the i -th dimension by

$$FE_i = \left(1 - \frac{\eta_i}{tp} \right) \quad (5.10)$$

where η_i represent the number of points inside the overlapping region and tp is the total number of sample points; as seen in Figure 5.6(a). Finally, we define $FE = \max(\{FE_i\})$ with $i = [1, m]$ for any given problem with m dimensions.

CLASS DISTANCE RATIO (CDR) compares the dispersion within the classes to the gap between the classes (Ho and Basu, 2002). For each data sample, compute the Euclidean distance to its nearest neighbor within the class (intra-class distance) and nearest-neighbor from the other class (inter-class distance), as shown in Figure 5.6(b). The CDR is the ratio of the averages of all intra-class and inter-class distances.

VOLUME OF OVERLAP REGION (VOR) provides an estimate of the amount of overlap between both classes in feature dimension space (Ho and Basu, 2002). The VOR is computed by finding, for each dimension, the maximum and minimum value of each class and then calculating the length of the overlap region. The length obtained from each dimension is then multiplied to measure the overlapping region, as depicted in Figure 5.6(c). The VOR is zero when there is at least one dimension in which the two classes do not overlap.

CANBERRA DISTANCE (CD) provides a numerical measure of the distance between pairs of points in a vector space. Suppose a problem has m feature dimensions, we take a rank statistic of the samples of each class, call it v_i for class 1 and v_i for class 2, for the i -th dimension. This produces two vectors \vec{v} and \vec{v} , such that $\vec{v} = (v_1, \dots, v_m)$ and $\vec{v} = (v_1, \dots, v_m)$. The CD is given by

$$CD(\vec{v}, \vec{v}) = \frac{1}{m} \sum_{i=1}^m \frac{|v_i - v_i|}{|v_i| + |v_i|}. \quad (5.11)$$

In this work, we use the CD to describe the distance between both classes using three rank statistics: (1) CD-1 uses the 1st quartile; (2) CD-2 uses the median; and (3) CD-3 uses the 3rd quartile.

The set of descriptive measures discussed above helps to minimize the information about each problem. Now, analyzing the algorithmic

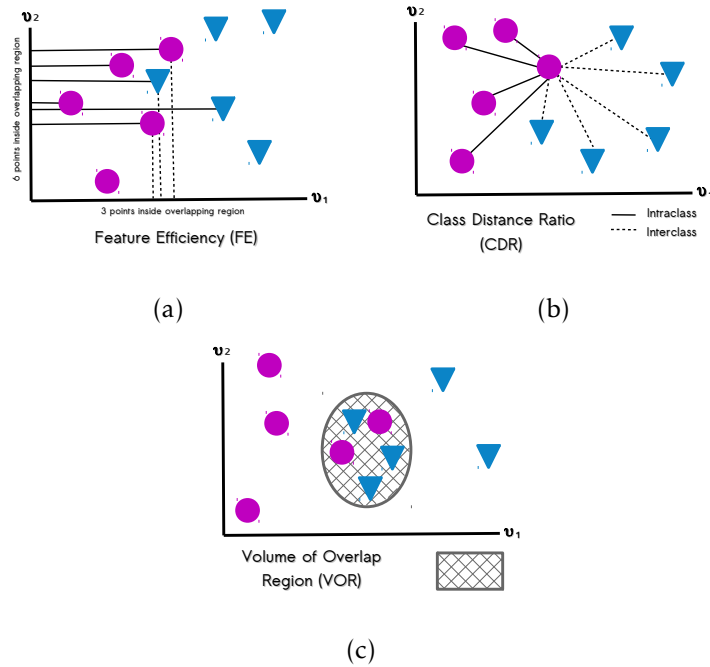


Figure 5.6: These figures depict the complexity features used to describe each classification problem as suggested in Ho and Basu (2002), where: (a) Feature Efficiency (FE); (b) Class Distance Ratio (CDR); and (c) Volume of Overlap Region (VOR).

complexity (big O notation) of the measures, these do not represent a significant computational cost. For instance, the FE, VOR, CD-1, CD-2 and CD-3 features mainly depend on a sorting process, which can have a complexity of $O(n \log n)$ where n is the number of instances of the problem. Moreover, the SD relies on computing the covariance matrix of the data which has a complexity of $O(n^2)$. Similarly, to compute the CDR feature we need to do all pairwise comparisons, which also has a complexity of $O(n^2)$.

Figure 5.7 provides a visual description of the descriptive power of each feature. The figure shows scatter plots where each point corresponds to a single problem $p \in Q'$, the x-axis is a particular feature (SD,

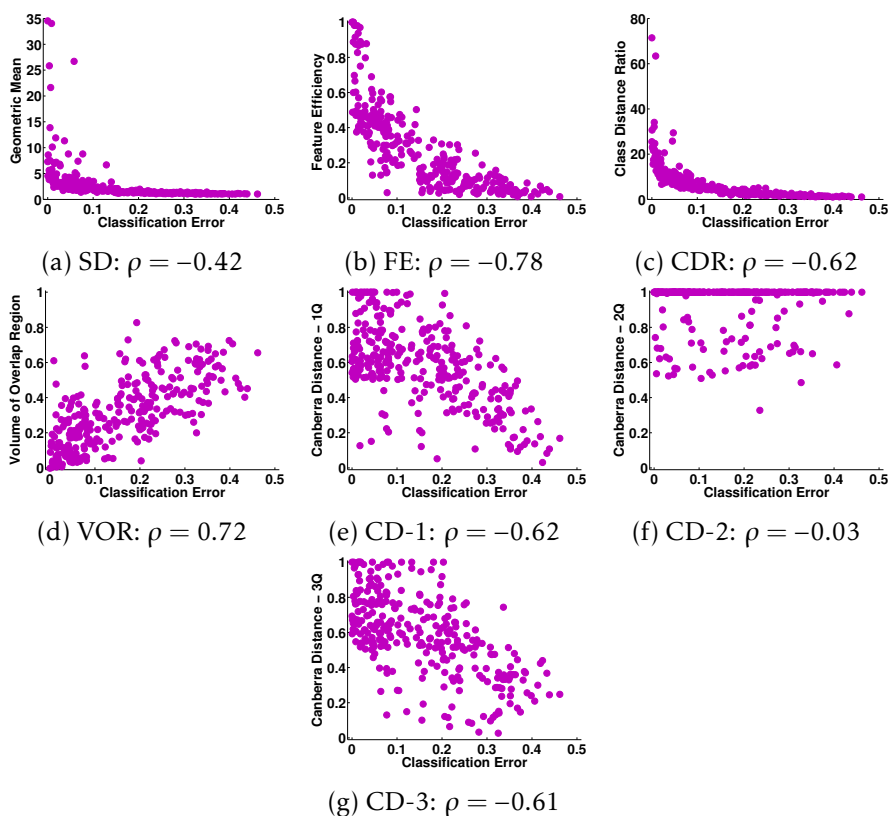


Figure 5.7: Scatter plots show the relationship between the CE_{μ} (x-axis) and each descriptive feature (y-axis) for all problems $p \in Q'$, where ρ specifies Pearson’s correlation coefficient.

FE, CDR, VOR, CD-1, CD-2 and CD-3) and the y-axis is the associated CE_{μ} . The legend of each plot also gives the Pearson’s correlation coefficient ρ . It is evident that all of the chosen features are correlated with PGPC performance, in particular FE, VOR, CDR, CD-1 and CD-3 show the highest correlation.

5.5 SUPERVISED LEARNING OF PEP MODELS

It is now possible to pose a symbolic regression problem using the set $T = \{(\beta_i, CE\mu_i)\}$ with $i = 1, \dots, |\mathcal{Q}'|$, where the goal is to evolve a model K that can predict each $CE\mu_i$ using β_i as input. Previous works have used several types of linear models (Graff and Poli, 2011; Martínez et al., 2012; Trujillo et al., 2011a, 2012, 2011b,c), but (Trujillo et al., 2011a,b,c) showed that non-linear models evolved with GP achieved higher prediction accuracy.

Therefore, in this work we use a tree-based GP, configured with the parameters given in Table 5.2. Three versions of the problem are posed, each with a different terminal set defined as subsets of all extracted features (4F, 5F, 7F) as specified in Table 5.3. Set 4F uses the features with the four highest correlation coefficients (FE, CDR, VOR and CD-1), set 5F uses the features with the five highest correlation coefficients (SD, FE, CDR, VOR and CD-1), and 7F uses all of the seven features. The function set is defined as $F = \{+, -, *, /, \sqrt{\cdot}, \sin, \cos, \log, x^y, |\cdot|, if\}$. Finally the fitness function is computed by the root mean squared error (RMSE) between the predicted CE and the true $CE\mu_i$, given by

$$f(K) = \sqrt{\frac{\sum_{i=1}^n (K(\beta_i) - CE\mu_i)^2}{n}}. \quad (5.12)$$

5.6 TESTING THE PEP MODELS

For each version of the symbolic regression problem defined above (with different feature sets), we performed 100 runs using two different test scenarios: (1) train and test the PEP models using only synthetic problems; and (2) train with synthetic problems and test with real-world problems. In the first scenario, we use 70% of the problems for training and the rest for testing, generating a random partition of the set of problems \mathcal{Q}' for each run. This is the simplest scenario, since both the training and testing problems are generated in the same manner.

PEP: PREDICTOR OF EXPECTED PERFORMANCE

Table 5.2: Parameters for the GP used to derive PEP models for PGPC algorithm.

| Parameter | Description |
|-------------------------------|--|
| <i>Population size</i> | 200 individuals |
| <i>Generations</i> | 100 generations |
| <i>Initialization</i> | <i>Ramped half-and-half,</i> with 6 levels of maximum depth |
| <i>Operator probabilities</i> | Crossover $p_c = 0.8$; Mutation $p_\mu = 0.2$ |
| <i>Hard maximum depth</i> | 12 levels |
| <i>Selection</i> | Tournament, size 3 |
| <i>Survival</i> | Keep best elitism |
| <i>Runs</i> | 100 |

Table 5.3: Three different features sets used as terminal elements for the symbolic regression GP algorithm.

| <i>Feature vector β</i> | |
|--|---------------------------------------|
| 4F | FE, CDR, VOR and CD-1 |
| 5F | SD, FE, CDR, VOR and CD-1 |
| 7F | SD, FE, CDR, VOR, CD-1, CD-2 and CD-3 |

In the second scenario, we test the PEP models trained with synthetic problems and evaluate their predictions on many real-world datasets, a more challenging scenario since the real-world problems have high dimensional data, imbalanced classes and different data distributions.

5.6.1 Testing on Synthetic Classification Problems

Table 5.4 summarizes the performance of the evolved PEPs, showing the median of the RMSE of the best solution found in each run for the training and testing sets, as well as the RMSE and Pearson's correlation coefficient ρ of the best solution found. The table presents three rows of results, one for each feature set (PEP-4F, PEP-5F and PEP-7F).

Table 5.4: Prediction performance of the evolved PEPs applied on the synthetic problems using each feature set (4F, 5F and 7F, see Table 5.3). Performance is given based on the RMSE and Pearson’s correlation coefficient, with bold indicating the best performance.

| | median <i>training</i> RMSE | median <i>testing</i> RMSE | <i>best</i> RMSE | <i>best</i> correlation |
|---------------|--------------------------------|-------------------------------|------------------|-------------------------|
| <i>PEP-4F</i> | 0.0320 | 0.0375 | 0.0318 | 0.9634 |
| <i>PEP-5F</i> | 0.0317 | 0.0362 | 0.0295 | 0.9688 |
| <i>PEP-7F</i> | 0.0326 | 0.0364 | 0.0317 | 0.9636 |

The numerical results are encouraging, suggesting that the PEP models can accurately predict PGPC performance. Moreover, there is a very small difference between training and testing performance, suggesting that the PEP models are not overfitted.

Figure 5.8 shows plots in three rows, where in each row we plot each feature set (PEP-4F, PEP-5F and PEP-7F). The plots on the left-hand side column show the PCE of the best PEP model and the true $CE\mu$ for all synthetic problems, specifying the RMSE. The plots on the right-hand side column show the $CE\mu$ and PCE as scatter plots, specifying the Pearson’s correlation coefficient ρ . The evolved PEPs produce accurate predictions with all feature sets.

5.6.2 Testing on Real-World Classification Problems

This section presents the results of testing the best evolved PEPs to predict the testing error of PGPC on real-world problems. To this end, twenty-two real-world datasets are chosen from the University of California Irvine (UCI) machine learning repository (Lichman, 2013), summarized in Table 5.5. Since our PEPs only consider binary classification, we use these datasets to build 40 binary classification problems. The problems are summarized in Table 5.6, specifying the name of the dataset and the classes used to define each problem, the number of total samples and the imbalance percentage of each problem computed

PEP: PREDICTOR OF EXPECTED PERFORMANCE

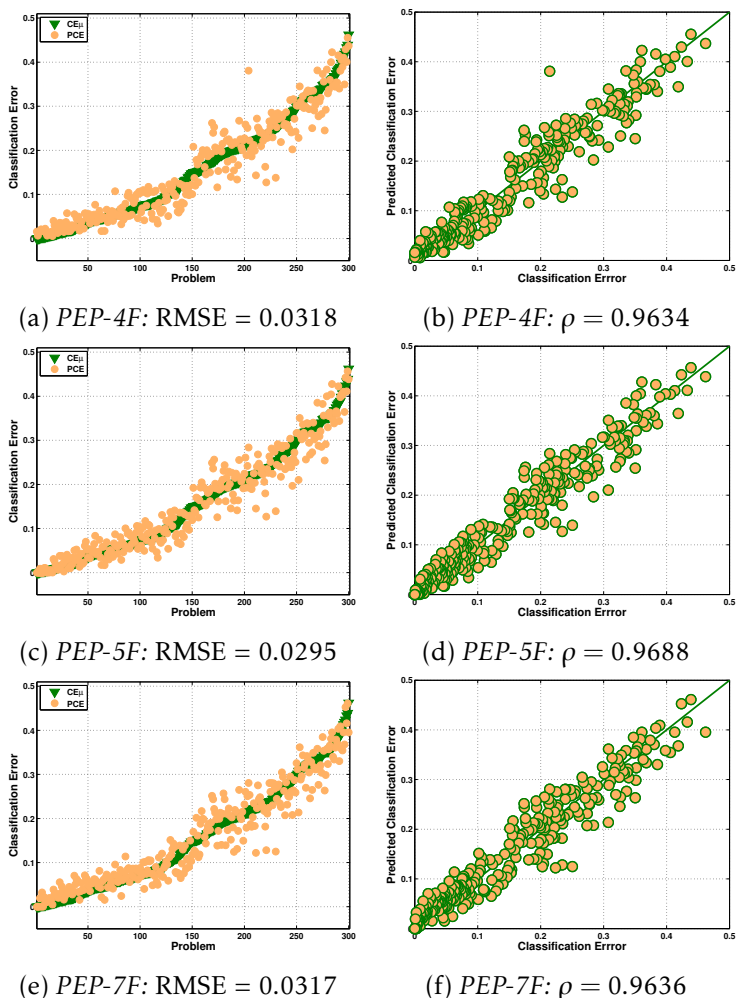


Figure 5.8: Figures show for synthetic problems, the performance prediction of the best PEP models evolved with the different feature set, each row belongs to each feature set: PEP-4F(top), PEP-5F(middle) and PEP-7F(bottom).

as $\frac{a-b}{c}$ where a and b are respectively the number of samples in the minority and majority class, and c is the total number of samples. Notice

Table 5.5: Real-world datasets from the UCI machine learning repository used in this work.

| No. | Problem | Classes | Dimensions | Description |
|-----|--------------------------------|---------|------------|---|
| 1 | <i>Balance scale</i> | 3 | 4 | Balance scale weight and distance database. |
| 2 | <i>Breast cancer wisconsin</i> | 2 | 8 | Original Wisconsin Breast Cancer Database. |
| 3 | <i>Breast tissue</i> | 6 | 9 | Dataset with electrical impedance measurements of freshly excised tissue samples from the breast. |
| 4 | <i>Cardiotocography</i> | 3 | 23 | Fetal cardiotocograms (CTGs) were automatically processed and the respective diagnostic features measured. |
| 5 | <i>EEG eye state</i> | 2 | 15 | All data is from one continuous EEG measurement with the Emotiv EEG Neuroheadset. |
| 6 | <i>Fertility</i> | 2 | 10 | 100 volunteers provide a semen sample analyzed according to the WHO 2010 criteria. |
| 7 | <i>Glass</i> | 6 | 10 | From USA Forensic Science Service; 6 types of glass. |
| 8 | <i>Indian liver patient</i> | 2 | 32 | This data set contains 416 liver patient records and 167 non liver patient records. |
| 9 | <i>Ionosphere</i> | 2 | 32 | Classification of radar returns from the ionosphere. |
| 10 | <i>Iris</i> | 3 | 4 | The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. |
| 11 | <i>Parkinsons</i> | 2 | 22 | Oxford Parkinson's Disease Detection Dataset. |
| 12 | <i>Pima indians diabetes</i> | 2 | 8 | From National Institute of Diabetes and Digestive and Kidney Diseases. |
| 13 | <i>Retinopathy</i> | 2 | 19 | This dataset contains features extracted from the Messidor image set to predict whether an image contains signs of diabetic retinopathy or not. |
| 14 | <i>Red wine</i> | 6 | 11 | The goal is to model wine quality based on physicochemical tests. |
| 15 | <i>Seed</i> | 3 | 7 | The examined group comprised kernels belonging to three different varieties of wheat. |
| 16 | <i>Sonarall</i> | 2 | 60 | The task is to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock. |
| 17 | <i>Tae</i> | 3 | 5 | The data consist of evaluations of teaching performance; scores are "low", "medium", or "high". |
| 18 | <i>Vertebral-column 2C</i> | 2 | 6 | Biomedical data set built by Dr. Henrique da Mota. |
| 19 | <i>Vertebral-column 3C</i> | 3 | 6 | Biomedical data set built by Dr. Henrique da Mota. |
| 20 | <i>White wine</i> | 6 | 11 | The goal is to model wine quality based on physicochemical tests. |
| 21 | <i>Wine</i> | 3 | 13 | Using chemical analysis determine the origin of wines. |
| 22 | <i>Zoo</i> | 7 | 3 | Artificial, 7 classes of animals. |

that the synthetic problems used to train the PEPs are completely balanced and relatively small problems in terms of number of samples, while the real-world problems are considerably more varied. In particular, considering class imbalance Figure 5.9 shows an histogram of the number of problems with different amounts of imbalance percentage.

Before testing the evolved PEP models, we compute the $CE\mu$ achieved by PGPC using 30 independent runs. PGPC performance is

PEP: PREDICTOR OF EXPECTED PERFORMANCE

Table 5.6: The 40 real-world binary classification problems based on the UCI datasets.

| No. | Problem | Classes | Instances | Imbalance % |
|-----|--------------------------------|---------|-----------|-------------|
| 1 | <i>Balance scale</i> | 1-3 | 576 | 0 |
| 2 | <i>Breast cancer wisconsin</i> | 1-2 | 699 | 31 |
| 3 | <i>Breast tissue</i> | 1-2 | 36 | 17 |
| 4 | <i>Breast tissue</i> | 1-3 | 39 | 8 |
| 5 | <i>Breast tissue</i> | 1-4 | 37 | 14 |
| 6 | <i>Breast tissue</i> | 2-3 | 33 | 9 |
| 7 | <i>Breast tissue</i> | 2-4 | 31 | 3 |
| 8 | <i>Breast tissue</i> | 3-4 | 34 | 6 |
| 9 | <i>Cardiotocography</i> | 1-2 | 1950 | 70 |
| 10 | <i>Cardiotocography</i> | 1-3 | 1831 | 81 |
| 11 | <i>Cardiotocography</i> | 2-3 | 471 | 26 |
| 12 | <i>EEG eye state</i> | 1-2 | 8388 | 17 |
| 13 | <i>Fertility</i> | 1-2 | 100 | 76 |
| 14 | <i>Glass</i> | 1-2 | 146 | 4 |
| 15 | <i>Glass</i> | 1-6 | 99 | 41 |
| 16 | <i>Glass</i> | 2-6 | 105 | 45 |
| 17 | <i>Indian liver patient</i> | 1-2 | 579 | 43 |
| 18 | <i>Ionosphere</i> | 1-2 | 351 | 28 |
| 19 | <i>Iris</i> | 1-2 | 100 | 0 |
| 20 | <i>Iris</i> | 1-3 | 100 | 0 |
| 21 | <i>Iris</i> | 2-3 | 100 | 0 |
| 22 | <i>Parkinsons</i> | 1-2 | 195 | 51 |
| 23 | <i>Pima indians diabetes</i> | 1-2 | 768 | 30 |
| 24 | <i>Red wine</i> | 5-6 | 1319 | 3 |
| 25 | <i>Retinopathy</i> | 1-2 | 1151 | 6 |
| 26 | <i>Seeds</i> | 1-2 | 140 | 0 |
| 27 | <i>Seeds</i> | 1-3 | 140 | 0 |
| 28 | <i>Seeds</i> | 2-3 | 140 | 0 |
| 29 | <i>Sonarall</i> | 1-2 | 208 | 7 |
| 30 | <i>Tae</i> | 1-2 | 99 | 1 |
| 31 | <i>Tae</i> | 1-3 | 101 | 3 |
| 32 | <i>Tae</i> | 2-3 | 102 | 2 |
| 33 | <i>Vertebral column 2C</i> | 1-2 | 310 | 35 |
| 34 | <i>Vertebral column 3C</i> | 1-2 | 210 | 43 |
| 35 | <i>Vertebral column 3C</i> | 1-3 | 160 | 25 |
| 36 | <i>Vertebral column 3C</i> | 2-3 | 250 | 20 |
| 37 | <i>White wine</i> | 5-6 | 3655 | 20 |
| 38 | <i>Wine</i> | 1-2 | 130 | 9 |
| 39 | <i>Wine</i> | 1-3 | 107 | 10 |
| 40 | <i>Zoo</i> | 1-2 | 61 | 34 |

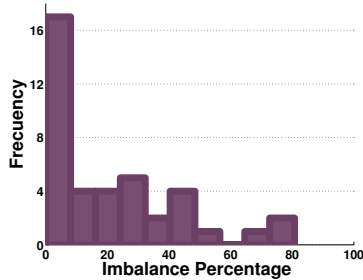


Figure 5.9: Histogram of imbalance percentage for the 40 real-world classification problems.

IMBALANCED PROBLEMS

1. The PEP models were trained with balanced problems.
2. The real-world test problems show a varied amount of imbalanced cases.

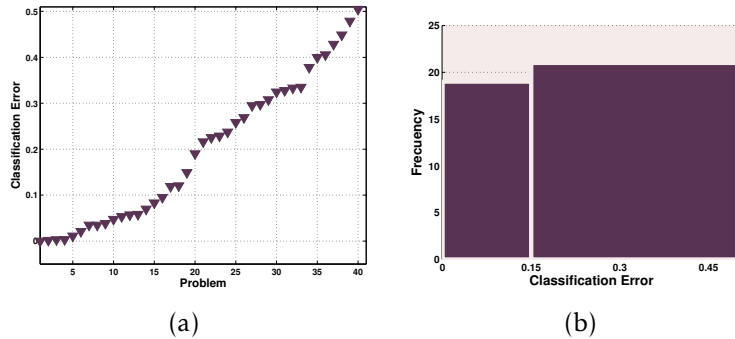


Figure 5.10: Performance of PGPC on the 40 real-world classification problems; where: (a) shows the CE_{μ} for each problem; and (b) shows the histogram over CE_{μ} .

summarized in Figure 5.10, showing: (a) the CE_{μ} for each problem and (b) the histogram over CE_{μ} . Figure 5.11 presents scatter plots of each descriptive feature (x -axis) and the CE_{μ} (y -axis) of each problem, specifying the corresponding Pearson's correlation coefficient ρ in the legend of each plot. The figures show that the best correlated features with CE_{μ} are FE and CD-1, respectively with ρ values of -0.73 and -0.71 . The rest of the features do not show particularly good correlation values, with SD clearly being the worst.

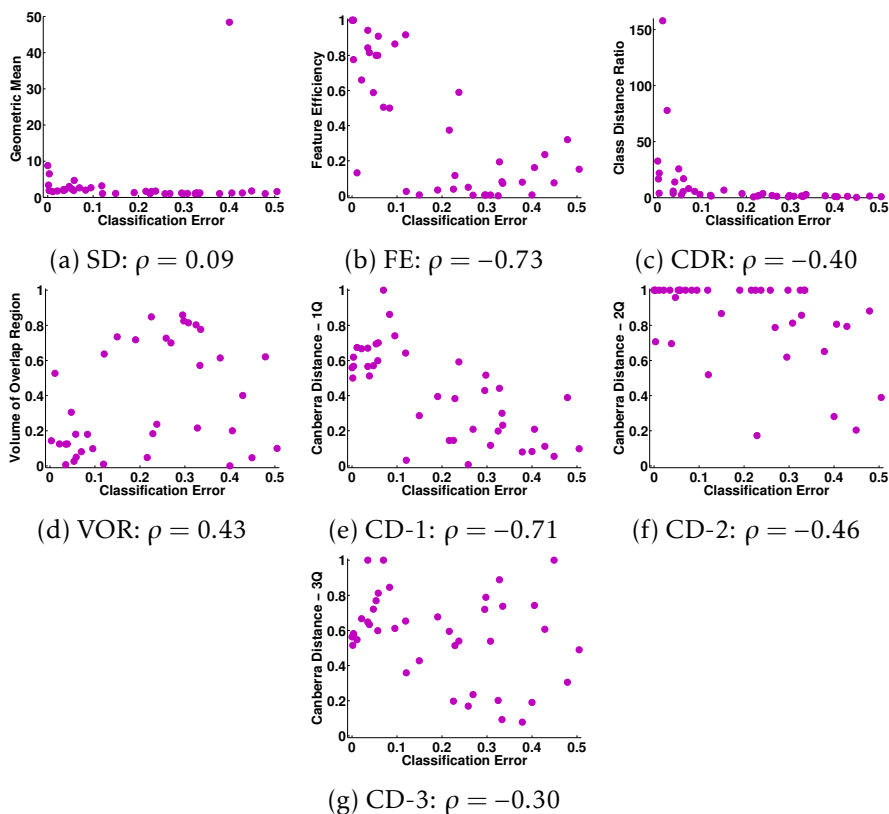


Figure 5.11: Scatter plots show for the real-world problems the relationship between the $CE\mu$ (x-axis) and each descriptive feature (y-axis). The legend specifies Pearson’s correlation coefficient ρ .

These results are different to what was observed on the synthetic problems. While VOR, CDR and CD-3 showed absolute correlation values above 0.6 on synthetic datasets, they were all below 0.44 on the real-world problems. This difference was particularly marked for SD, on synthetic problems the correlation coefficient was -0.42 but on real-world problems it is 0.09 . In fact, only FE and CD-1 showed a good correlation on both sets.

Table 5.7: Prediction performance of the evolved PEPs applied on the real-world problems using each feature set (4F, 5F and 7F, see Table 5.3). Performance is given based on the RMSE and Pearson’s correlation coefficient, with bold indicating the best performance.

| | <i>median</i> RMSE | <i>best</i> RMSE | <i>best</i> correlation |
|---------------|--------------------|------------------|-------------------------|
| <i>PEP-4F</i> | 0.1522 | 0.0828 | 0.8634 |
| <i>PEP-5F</i> | 0.1583 | 0.0929 | 0.8823 |
| <i>PEP-7F</i> | 0.1676 | 0.0930 | 0.8046 |

Table 5.7 summarizes the performance of the evolved PEPs applied on the real-world problems, showing the median of the RMSE of the best solution found, as well as the RMSE and Pearson’s correlation coefficient ρ of the best solution. The table presents three rows of results, one for each feature set (PEP-4F, PEP-5F and PEP-7F). In this case, the best performance is achieved by PEP-4F, which was unexpected. However, if we contrast the results with those achieved on the set of synthetic problems, shown in Table 5.4, a performance drop-off is evident, based on both median and best performance.

Figure 5.12 shows three rows of plots, one for each feature set (PEP-4F, PEP-5F and PEP-7F). The figures on the left-hand side column show the PCE of the best PEP model and the true $CE\mu$ for all real-world problems, specifying the RMSE. The figures on the right-hand side column show the $CE\mu$ and PCE as scatter plots, specifying the Pearson’s correlation coefficient ρ . Again, these figures reveal that the evolved PEP models provide less accurate prediction on real-world problems.

5.7 COMPARATIVE STUDY OF EI AND PEP

FDC is a measure for problem difficulty originally proposed for genetic algorithms (GA’s) (Jones and Forrest, 1995) and later extended to GP (Tomassini et al., 2005). The logic behind FDC proceeds as follows. Assume that we can compute the genotypic distance between each valid individual and the (global) optimum to a problem. If this distance is

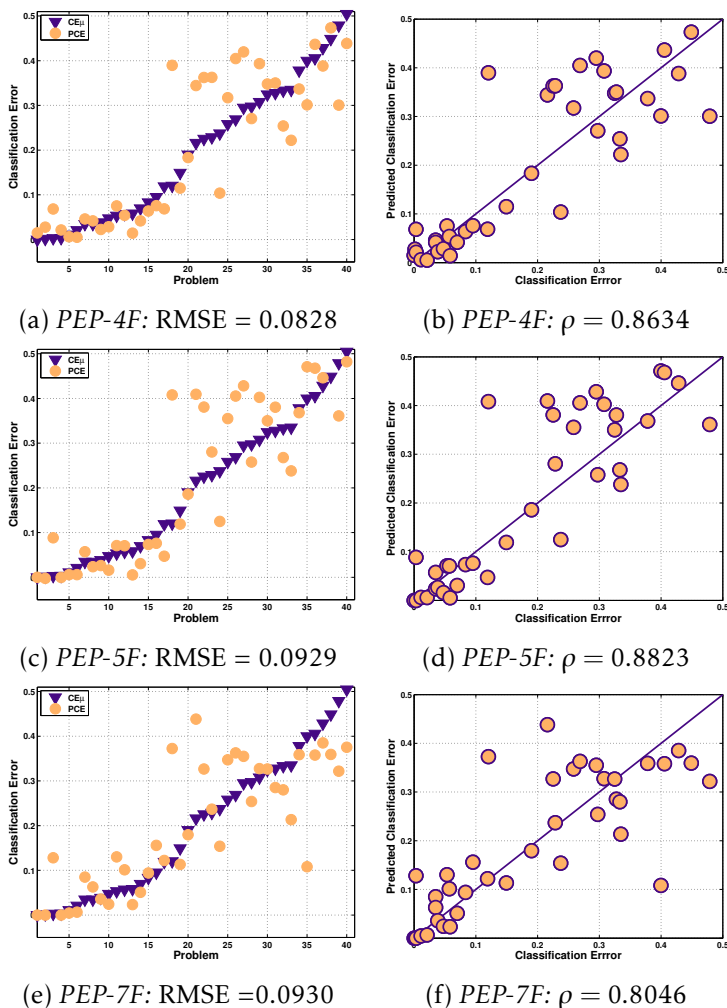


Figure 5.12: Figures show for real-world problems, the performance prediction of the best PEP models evolved with the different feature set, each row belongs to each feature set: PEP-4F(top), PEP-5F(middle) and PEP-7F(bottom).

negatively correlated with the fitness of each individual then the search problem should be characterized as *easy*, and it should be characterized

as difficult if no correlation is detected. Moreover, a problem should be considered to be deceptive if the correlation is positive. While FDC has shown to be reliable in many test cases, its more glaring weakness is that the optimal solution must be known *a priori*, somewhat not realistic for real-world problems.

Following the same general assumptions of FDC, Vanneschi et al. (2004) propose the NSC. In the case of NSC, knowledge about the global optimum is not required. Instead, NSC relies on the concept of *fitness clouds*, a scatter plot where for each genotype v a point is plotted on a 2-D plane, where the horizontal axis corresponds with the fitness of v given by $f(v)$, and the vertical axis represents the fitness $f(v)$ of a neighbouring genotype v . The hypothesis behind NSC is that the *fitness cloud shape* provides a meaningful description of the evolvability of a problem for GP-based search. The NSC is computed by assuming a piecewise linear relationship between $f(v)$ and $f(v)$ for a sample of ζ individual genotypes and computing the slope of the scatter points within a set of equally spaced segments of the $f(v)$ axis. In the original implementation, individuals are sampled using the Metropolis-Hastings algorithm, neighbours are generated using standard sub-tree mutation, and the representative neighbour v for each genotype v is chosen using tournament selection. The NSC is given in the range of $(-\infty, 0]$, where a value of 0 represents a highly evolvable (assumed to be easy) problem, and a negative NSC indicates a less evolvable (more difficult) problem.

One way to characterize problem difficulty is to attempt to predict the expected performance that a GP search will achieve on a given problem instance, a more direct approach. Following this line of thought, two approaches have been proposed in GP literature. We will use the proposal presented in Trujillo et al. (2011a,b,c), where is predict the performance of a GP-classifier using descriptive features that characterize the geometry of the data distribution in feature space. The PEPs where built using non-linear GP models.

After reviewing the basic methodology of EI's and PEP's, one practical and computational difference stands out. On the one hand, EI's use a very large sampling of the search space to derive an accurate measure.

In effect, this means that for every new problem instance it is necessary to perform this costly computational step. However, executing the actual GP search might in fact be faster. Therefore, the best use of EI's would be to characterize a whole class of problems, where the estimate of search difficulty provided by the EI could generalize to the entire class of problems.

On the other hand, a PEP model is used quite easily and directly for each new problem instance. Practically, the only possible bottleneck would be the computational cost of calculating the set of descriptive features for each problem. However, in order to learn a new PEP a very large number of experimental runs must be carried to derive the training data. Moreover, each PEP is strongly linked to a specific GP system and implementation, and even small deviations from the configuration of the GP system might cause the predictive model to break-down.

The goal of the experimental work is to evaluate and compare the predictive accuracy of a state-of-the-art EI (NSC) and a PEP model for a GP-based classifier PGPC. To this end, it is necessary:

1. Generate and solve with PGPC the set of synthetic classification problems using PGPC.
2. Calculate the NSC over all synthetic classification problems.
3. Build the PEP models following the methodology proposed in Trujillo et al. (2011b,c).
4. Analyze the results.

First, a large set of synthetic classification problems are generated and solved with PGPC, executing 30 independent runs on each problem and computing the average classification error as the estimate of the expected performance of PGPC. To evaluate the performance of PGPC, 300 two-class classification problems are randomly generated using Gaussian mixture models (GMM's), these conform set \mathcal{Q} . Then, for every problem $p \in \mathcal{Q}$ the average test error of PGPC is computed from 30 independent runs.

Second, to compute the NSC on each of the classification problems in set \mathcal{Q} , the approach is quite straightforward, since it is possible to directly apply the NSC algorithm to every problem. The algorithm described in Vanneschi et al. (2004) is used here with the same parameters except for the total amount of sampled individuals ζ . Whereas in Vanneschi et al. (2004) $\zeta = 40,000$, here $\zeta = 10,000$, a practical choice to reduce computation time; however, some informal tests showed that the results are consistently similar for both values in the group of experiments reported here. Figure 5.13 presents a scatter plot where the horizontal axis is the average classification error and the vertical axis is the NSC.

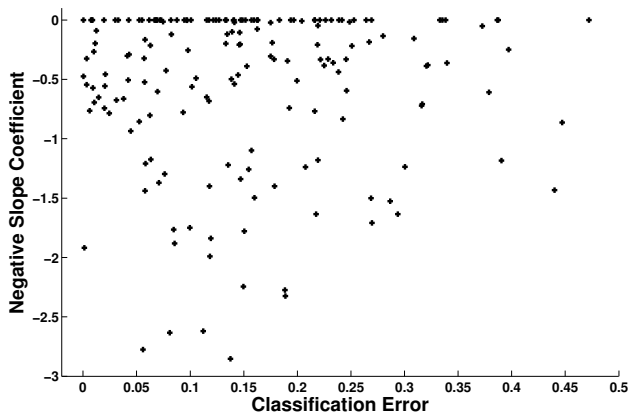


Figure 5.13: Scatter plot of the average classification error achieved by PGPC on each problem and the corresponding NSC value. Pearson's correlation coefficient $\rho = 0.02$.

The results clearly suggest that the NSC does not correlate with PGPC performance, in particular we can see how many problems are characterized as easy (with NSC equal or close to zero) even when the performance achieved by PGPC is quite poor. This suggests that an EI such as the NSC is limited as a predictor of GP performance since it only considers the frame of reference of the search process; i.e., it can only provide an approximate measure of the difficulty of the search but

tells us very little regarding the difficulty of the underlying problem that the GP is intended to solve.

Third, we show how a PEP estimates the performance of PGPC on the set of classification problems presented above. The PEP is derived following the approach described in Trujillo et al. (2011b). The symbolic regression models are tested derived with GP (GP-PEP), reported to achieve the best results in Trujillo et al. (2011b). In the case of the GP-PEP models, the problem descriptors are used as terminal elements

$$T = \{SD, VOR, FE, CDR\} \quad (5.13)$$

while the function set is defined as

$$F = \{+, -, *, /, \sqrt{\cdot}, \sin, \cos, \log, x^y, |\cdot|, if\} \quad (5.14)$$

Moreover, fitness is computed by the RMSE calculated on a set of n training problems, given by

$$f(K) = \sqrt{\frac{\sum_{i=1}^n (K(\beta_i) - \epsilon_i)^2}{n}}. \quad (5.15)$$

where β_i is the vector of descriptive features and ϵ_i is the performance estimate on a training problem i . Finally, the GP is executed for the set \mathcal{Q} of classification problems which is divided into a training set and a testing set and 30 runs are executed with different random partitions.

In general, the PEP model exhibits a very good predictions, with the median error around 5 and 7 percentage points of classification accuracy. Another look at the predictive accuracy of the PEP models is shown in the scatter plots of Figure 5.14. In these plots the predictive classification error is plotted against the average error of PGPC on each problem, using the best GP-PEP model. It is clear that the prediction of the PEP models is strongly correlated with the average performance of PGPC, and the Pearson's correlation coefficient presented with each plot confirms this observation.

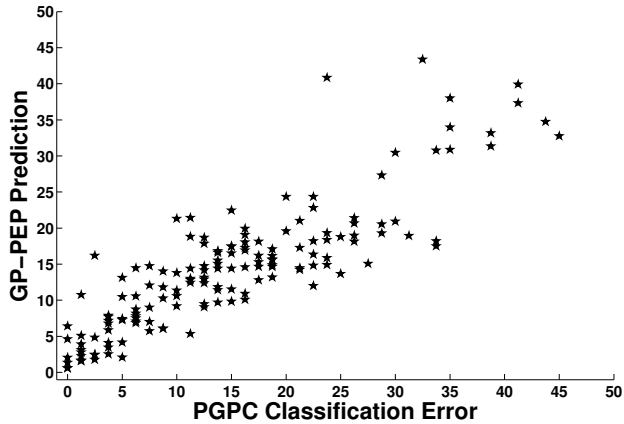


Figure 5.14: Scatter plots that show the average performance of PGPC (x-axis) and the predicted performance of each PEP model (y-axis). The legend specifies Pearson's correlation coefficient $\rho = 0.86$.

Finally, two groups of problem difficulty prediction tools are analyzed, named Evolvability Indicators and Predictors of Expected Performance. The former group of measures attempt to capture how amenable the fitness landscape is to a GP, whereas the latter groups takes as input a set of descriptive features of a problem and produces as output an estimate of the expected performance of the GP search. The key lessons of this study are the following. Firstly, while EIs (the Negative Slope Coefficient is considered) can give a good estimation of the difficulty of the search problem, they are not necessarily correlated with expected performance; Secondly, the results suggest that PEPs achieve a highly accurate prediction of GP performance.

PEP: PREDICTOR OF EXPECTED PERFORMANCE

6

SPEP: SPECIALIST PREDICTORS OF EXPECTED PERFORMANCE

The above results are encouraging, but for a real-world application even small improvements in the quality of the predictions could have non-negligible effects. Therefore, in this section we propose an ensemble approach using several PEP models, each one referred to as an SPEP. We propose an ensemble approach for two main reasons. First, previous works suggest that ensemble-based modeling can improve performance in a variety of scenarios (Folino et al., 2010; Zhou, 2012). Second, an ensemble approach allows us to obtain two types of predictions, a numerical prediction of expected performance and a categorical or fuzzy prediction based on the chosen ensemble component used to compute the final prediction. The proposal is depicted in Figure 6.1, an extension of the basic PEP approach shown in Figure 5.1. However, in the SPEP approach before computing the PCE for a given problem, each problem is classified into a specific group using its corresponding feature vector β . Each group is associated to a particular SPEP in the ensemble, hence if a problem is classified into the i -th group then the i -th SPEP is used to compute the predicted PGPC performance on the test set.

To implement this approach, several design choices must be specified. First, how to define a meaningful grouping of problems. Second, train SPEPs that are specialized for each group in order to build the ensemble. Third, chose the correct SPEP for a particular problem by determining its group membership. Each of these issues are discussed next.

SPEP: SPECIALIST PREDICTORS OF EXPECTED PERFORMANCE

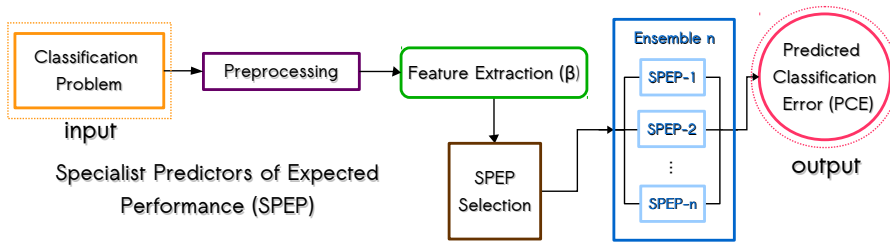


Figure 6.1: Block diagram showing the proposed SPEP approach. The proposed approach is an extension of the basic PEP approach of Figure 5.1, with the additional ensemble approach, where problems are first classified into prespecified groups and based on this a corresponding specialized model (SPEP) is chosen to compute the PCE of PGPC on the test set.

6.1 GROUPING PROBLEMS BASED ON PGPC PERFORMANCE AND TRAINING SPEPs

The proposal is to group problems based on the performance of PGPC given by $CE\mu$. This can be seen as a categorical prediction, where problems are grouped into general groups of different difficulty; e.g. easy and hard problems. In particular, we propose two different groupings based on $CE\mu$, using either two or three groups as shown in Figure 6.2. The groups were chosen in such a way that the number of (synthetic) problems in each group would be approximately the same, in this way posing a balanced classification task for the SPEP approach. Figure 6.2 shows the ranges of PGPC performance for each group and the number of synthetic problems (Figure 6.2(a)) and real-world problems (Figure 6.2(b)) that fall within each group. The plots on the top divide the set of problems into two groups, while the plots on the bottom divide the set of problems into three. Finally, for clarity, since the two group division requires two SPEPs, we refer to a solution for this task as an Ensemble-2, while a solution for the three group task is referred to as an Ensemble-3.

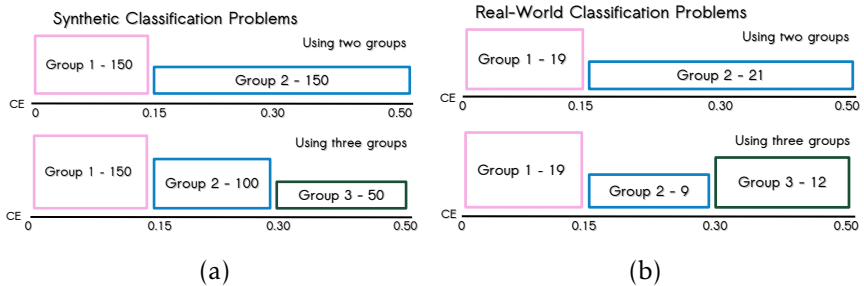


Figure 6.2: The proposed groupings of classification problems used with the SPEP approach, showing the ranges of PGPC performance and the number of problems in each group.

For each group an SPEP is trained using the strategy described in the previous section for PEPs. Except that instead of using all of the synthetic problems, each SPEP is trained using the subset of synthetic problems from the corresponding group, as depicted in Figure 6.2. Since we are interested in presenting the best possible prediction of PGPC performance on real-world problems, we must select the best predictive models. Therefore, the testing phase is performed using two subsets of the real-world problems, one for validation and other for testing.

6.2 SPEP SELECTION

As depicted in Figure 6.1, in order to choose an SPEP we must first classify each problem to its corresponding group. This is a straightforward classification task, solved using each problem's feature vector β as the decision variables. Several classification algorithms are tested (Duda et al., 2000), namely:

1. Euclidean distance classifier (EDC).
2. Mahalanobis distance classifier (MDC).
3. Naive Bayes classifier (NBC).

SPEP: SPECIALIST PREDICTORS OF EXPECTED PERFORMANCE

4. Support Vector Machine (SVM), with Gaussian radial basis function kernel and a default scaling factor of 1.
5. K-Nearest Neighbor (KNN), using $K = 5$ neighbors.
6. Treebagger Classifier (TBC), using 3 trees.
7. Probabilistic Genetic Programming Classifier (PGPC), parameters on Table 5.1.

Moreover, the classification task is posed using different subsets of the features in β as previously described in Table 5.3. We apply all classifiers using all subsets of features on both the two-group and three-group classification tasks.

As done for the SPEP models, in all cases the complete set of synthetic problems \mathcal{Q}' is used to train the classifiers. The testing phase is performed with two sets, 10% of the real-world problems are used as a validation set while the remaining 90% of real-world problems are used for testing. After performing 100 independent runs, the best solution is chosen based on its validation set performance, and methods are compared based on the performance on the testing set. If several solutions achieve the best validation set performance, than the final solution used in the ensemble is randomly chosen.

6.3 EVALUATION OF SPEP ENSEMBLES

This section presents the performance of the evolved SPEP models, and the performance of the complete ensembles, using both the true problem groups (an oracle approach, where the correct SPEP is always chosen) and the predicted group by the best classifier (a more realistic testing scenario).

6.3.1 *Ensemble-2 Solutions*

To visualize the underlying difficulty of choosing the correct SPEP for a given problem (i.e., determining the group to which it belongs to)

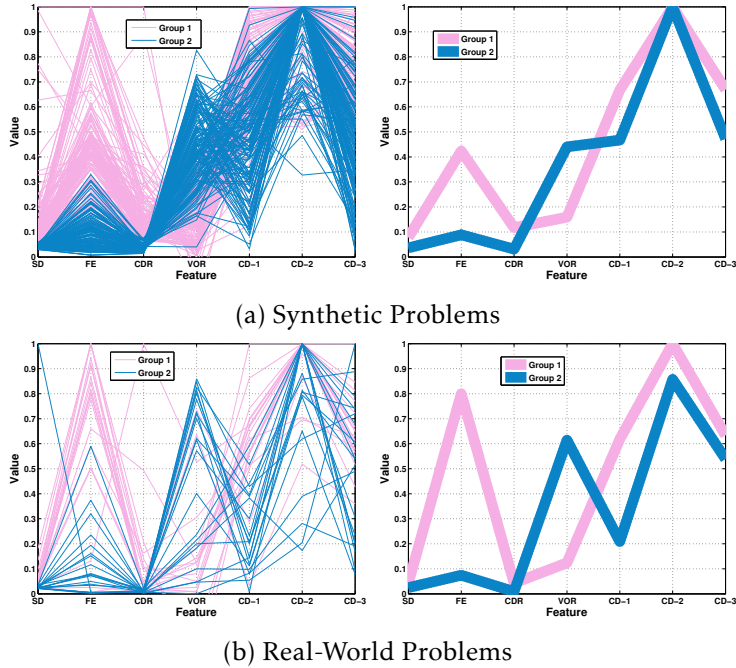


Figure 6.3: Parallel coordinate plots dividing the set of problems into two groups, where each coordinate is given by a feature in β . Plots are shown for synthetic (a) and real-world problems (b). The plots on the left show a single line for each problem, while the plots on the right show the median values for each group. For clarity in the parallel plots, the features SD and CDR were rescaled to values between $[0, 1]$.

Figure 6.3 presents a parallel coordinate plot dividing the set of problems into two groups, where each coordinate is given by a feature in β . Plots are shown for synthetic (Figure 6.3(a)) and real-world problems (Figure 6.3(b)). The plots on the left show a single line for each problem, while the plots on the right show the median values for each group. For clarity in the parallel plots, the features SD and CDR were rescaled to values between $[0, 1]$.

Table 6.1 summarizes the performance of the best SPEP models used to build the Ensemble-2 solution. The first column specifies the

SPEP: SPECIALIST PREDICTORS OF EXPECTED PERFORMANCE

Table 6.1: RMSE of the best evolved SPEP models, using different feature sets (first column). Performance is given based on training and testing set. Moreover, each SPEP- i corresponds to the i -th problem group but is tested on both problem groups, as specified in the fourth column. Bold indicates the best performance on each group.

| | <i>SPEP</i> | <i>training</i> | <i>Testing group</i> | <i>testing</i> |
|----|-------------|-----------------|----------------------|----------------|
| 4F | SPEP-1 | 0.0201 | 1 | 0.0315 |
| | | | 2 | 0.2470 |
| | SPEP-2 | 0.0341 | 1 | 0.1445 |
| | | | 2 | 0.0919 |
| 5F | SPEP-1 | 0.0195 | 1 | 0.0380 |
| | | | 2 | 0.1819 |
| | SPEP-2 | 0.0380 | 1 | 0.1119 |
| | | | 2 | 0.0832 |
| 7F | SPEP-1 | 0.0212 | 1 | 0.0469 |
| | | | 2 | 0.2096 |
| | SPEP-2 | 0.0332 | 1 | 0.1586 |
| | | | 2 | 0.1014 |

feature subset used from β . The second column specifies the evaluated SPEP, SPEP-1 was trained with synthetic problems from the first group while SPEP-2 was trained with problems from the second group. The training RMSE is given in column 3. Every SPEP was tested on real-world problems from both groups, to illustrate the performance difference and specialization of each model; this is specified in the fourth column. The final column gives the testing RMSE on each group.

The results show that the SPEP models are specialized to their groups, achieving error values below 0.1 when tested using problems from their groups, while performing worse when tested on problems from the other group. In general, performance on testing set is good, particularly if we compare with the results achieved by the PEP models from the preceding section. Finally, performance is similar for all feature sets when considering testing performance, with the best performance on Group 1 achieved by using the set 4F and the best performance on Group 2 with set 5F.

Table 6.2: Performance on the SPEP selection problem for all tested classifiers, showing the median classification error from 100 independent runs. The performance is given on the training and testing sets. Bold text indicates the best performance on each feature set.

| <i>Algorithm</i> | | EDC | MDC | NBC | SVM | KNN | TBC | PGPC |
|------------------|-----------------|--------|--------|--------|--------|--------|--------|---------------|
| 4F | <i>training</i> | 0.1533 | 0.0567 | 0.0200 | 0.0233 | 0.0133 | 0.0067 | 0.0100 |
| | <i>testing</i> | 0.2500 | 0.1389 | 0.1111 | 0.1111 | 0.1389 | 0.1111 | 0.0833 |
| 5F | <i>training</i> | 0.1533 | 0.0567 | 0.0200 | 0.0200 | 0.0200 | 0.0067 | 0.0100 |
| | <i>testing</i> | 0.2778 | 0.1389 | 0.1389 | 0.1389 | 0.1667 | 0.1389 | 0.1111 |
| 7F | <i>training</i> | 0.1533 | 0.0467 | 0.0200 | 0.0033 | 0.0200 | 0.0067 | 0.0100 |
| | <i>testing</i> | 0.2778 | 0.1389 | 0.1389 | 0.2500 | 0.1667 | 0.1111 | 0.0972 |

Table 6.3: Performance on the SPEP selection problem for all tested classifiers, showing the classification error of the best solution found, evaluated over all real-world problems, with bold indicating the best performance on each feature set.

| Feature Set | EDC | MDC | NBC | SVM | KNN | TBC | PGPC |
|-------------|--------|--------|--------|--------|--------|--------|---------------|
| 4F | 0.2500 | 0.1250 | 0.1000 | 0.1000 | 0.1250 | 0.1250 | 0.0500 |
| 5F | 0.2750 | 0.1250 | 0.1250 | 0.1250 | 0.1500 | 0.1250 | 0.1000 |
| 7F | 0.2750 | 0.1250 | 0.1250 | 0.2500 | 0.1500 | 0.1250 | 0.0250 |

The results in Table 6.1 represent the best possible performance if the correct problem group is chosen, but also confirm that if the correct group is not chosen than prediction accuracy can decline. Table 6.2 summarizes the performance of all of the tested classifiers for the two-group case, showing the median classification error achieved on the training and testing sets. On these tests, PGPC achieves the best performance based on test error.

Table 6.3 shows the performance of the best classifier obtained from each method and chosen based on the validation set. In this table performance is given using all real-world problems. Again, PGPC clearly outperforms all other variants, with the best performance achieved using feature set 7F with a classification error of 0.0250.

SPEP: SPECIALIST PREDICTORS OF EXPECTED PERFORMANCE

Table 6.4: Ensemble-2 prediction accuracy using each feature set (4F, 5F and 7F), using the best evolved SPEPs and the best classifiers with each feature set. Performance is given based on the RMSE and Pearson’s correlation coefficient when evaluated on the synthetic and real-world problem sets; with bold indicating the best performance on real-world problems.

| Feature Set | Synthetic | | Real-world | |
|-------------|---------------|---------------|---------------|---------------|
| | RMSE | correlation | RMSE | correlation |
| 4F | 0.0284 | 0.9709 | 0.0818 | 0.8717 |
| 5F | 0.0302 | 0.9984 | 0.0736 | 0.8981 |
| 7F | 0.0276 | 0.9728 | 0.0897 | 0.8514 |

It is now possible to evaluate the performance of the complete Ensemble-2 solutions, using the best evolved SPEPs and the best classifier. These results are summarized in Table 6.4, specifying the RMSE and Pearson’s correlation coefficient when evaluated on the synthetic and real-world problem sets. These tests show that the Ensemble-2 solutions can achieve low predictive error and a high correlation with the true PGPC performance, for both synthetic and real-world problems. In particular, using feature set 5F correlation on synthetic problems is close to unity, while performance on the real-world problems show the lowest error and approximately 0.9 correlation.

Focusing on the real-world problems, Figure 6.4 summarizes the performance of the Ensemble-2 predictors using each feature set (each row of the figure). The column on the left-hand side shows plots of the ground truth $CE\mu$ of each problem (triangles) and the Ensemble-2 PCE. These plots show three types of PCE: (1) correctly classified problems for which the appropriate SPEP was selected (CC-PCE); (2) misclassified problems for which an incorrect SPEP was selected (MC-PCE); and (3) for the misclassified problems the oracle SPEP prediction (O-PCE), which is the PCE produced by the correct SPEP. The column on the right-hand side of Figure 6.4 presents scatter plots of the true $CE\mu$ and the PCE, using the same notation.

These plots provide a graphical confirmation of the quality of the performance prediction. It is important to highlight the impact of a misclassified problem, shown as a black circle, compared to the prediction on the same problem if the correct SPEP had been chosen (O-PCE). For all problems for which the correct SPEP was chosen the PCE is highly correlated with the ground truth with only marginal differences in most cases.

6.3.2 Ensemble-3 Solutions

Figure 6.5 presents a parallel coordinate plot dividing the set of problems into three groups, where each coordinate is given by a feature in β . Plots are shown for synthetic (Figure 6.5(a)) and real-world problems (Figure 6.5(b)). The plots on the left show a single line for each problem, while the plots on the right show the median values for each group. For clarity, features SD and CDR were rescaled to values between $[0, 1]$.

Table 6.5 summarizes the performance of the best SPEP models used to build the Ensemble-3 solution. The first column, specifies the feature subset used from β . The second column specifies the evaluated SPEP. SPEP-1 was trained with synthetic problems from the first group, SPEP-2 with problems from the second group and SPEP-3 with problems from the third group. The third column shows the training RMSE, the fourth column shows the testing group and the final columns shows the testing RMSE.

Again, the results show that the SPEP models are specialized to their respective groups. Performance on the testing set is better than the simple PEP models, but worse than the Ensemble-2 solution presented before. All feature sets produce similar performance on testing set problems, with the best performance on Group 1 and Group 2 achieved by using set 4F, and the best performance on Group 3 with set 5F.

The results summarized in Table 6.5 represent the best possible performance if the correct problem group is chosen, but also confirm that

SPEP: SPECIALIST PREDICTORS OF EXPECTED PERFORMANCE

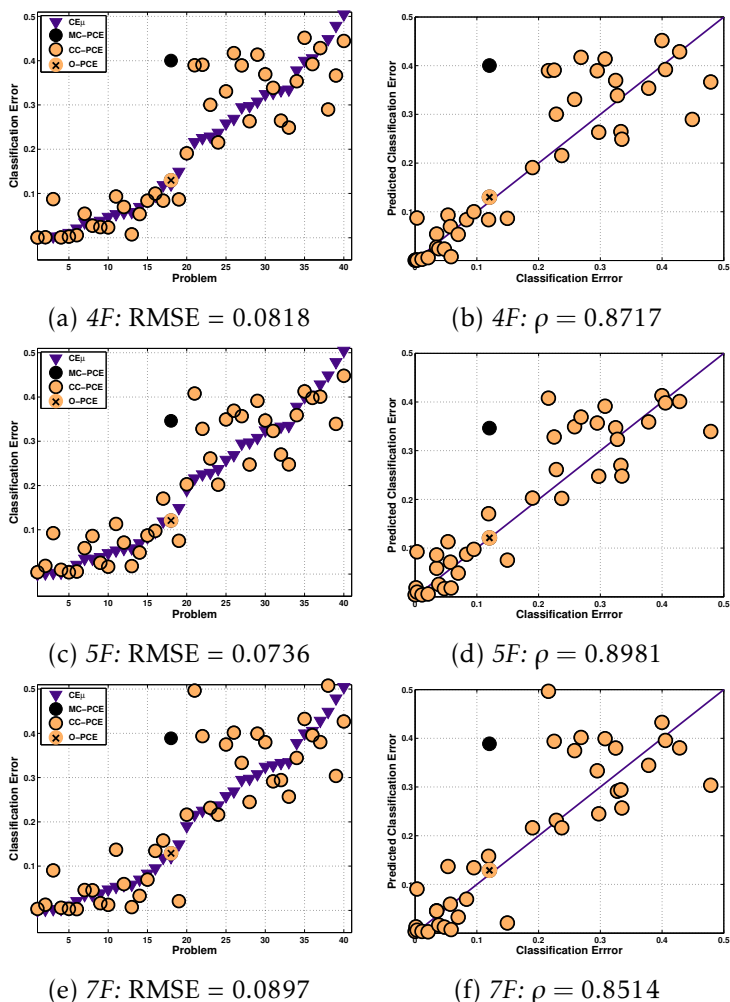
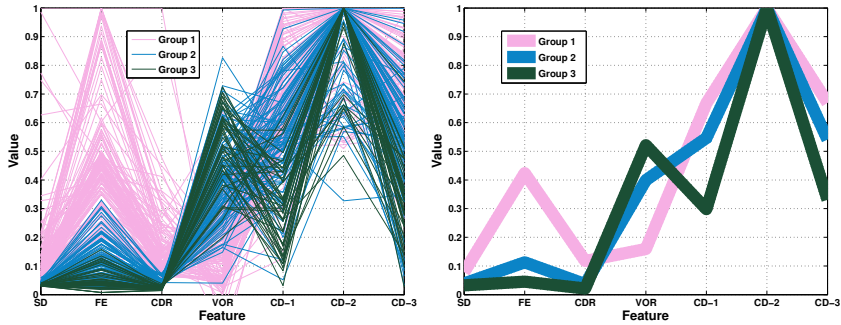
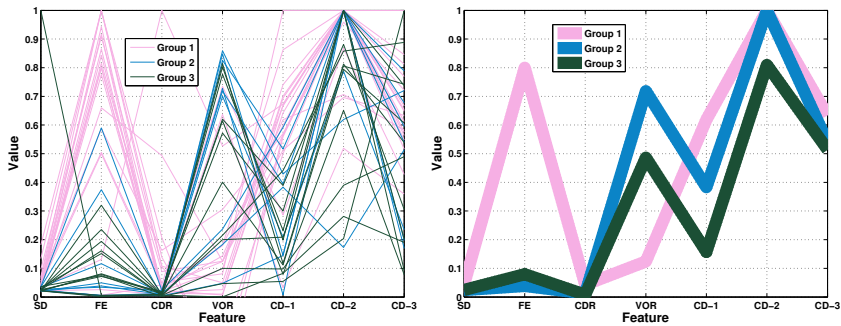


Figure 6.4: Performance prediction of the best Ensemble-2 solutions for each feature set: 4F(top), 5F(middle) and 7F (bottom).

if the correct group is not chosen than prediction accuracy can decline. Table 6.6 summarizes the performance of all of the tested classifiers for the three-group case, showing the median classification error achieved on the training and testing sets. On these tests, TBC achieves the best



(a) Synthetic Problems



(b) Real-World Problems

Figure 6.5: Parallel coordinate plots dividing the set of problems into three groups, where each coordinate is given by a feature in β . Plots are shown for synthetic (a) and real-world problems (b). The plots on the left show a single line for each problem, while the plots on the right show the median values for each group. For clarity in the parallel plots, the features SD and CDR were rescaled to values between $[0, 1]$.

median performance. Table 6.7 focuses on the performance of the best classifier evaluated over all real-world problems. Again, TBC outperforms all other variants, with the best performance achieved using feature set 5F with a classification error of 0.1750.

It is now possible to evaluate the performance of the complete Ensemble-3 solutions, using the best evolved SPEPs and the best classifier. These results are summarized in Table 6.8, specifying the RMSE

SPEP: SPECIALIST PREDICTORS OF EXPECTED PERFORMANCE

Table 6.5: RMSE of the best evolved SPEP models, using different feature sets (first column). Performance is given based on training and testing set. Moreover, each SPEP- i corresponds to the i -th problem group but is tested on all problem groups, as specified in column 4. Bold text indicates best performance on each group.

| | <i>SPEP</i> | <i>training</i> | <i>Testing group</i> | <i>testing</i> |
|----|-------------|-----------------|----------------------|----------------|
| 4F | SPEP3-1 | 0.0201 | 1 | 0.0315 |
| | | | 2 | 0.1312 |
| | | | 3 | 0.2767 |
| | SPEP3-2 | 0.0303 | 1 | 0.1883 |
| | | | 2 | 0.0302 |
| | | | 3 | 0.1459 |
| | SPEP3-3 | 0.0264 | 1 | 0.3955 |
| | | | 2 | 0.1349 |
| | | | 3 | 0.0532 |
| 5F | SPEP3-1 | 0.0195 | 1 | 0.0380 |
| | | | 2 | 0.2076 |
| | | | 3 | 0.1602 |
| | SPEP3-2 | 0.0313 | 1 | 0.0931 |
| | | | 2 | 0.0380 |
| | | | 3 | 0.1245 |
| | SPEP3-3 | 0.0294 | 1 | 0.2691 |
| | | | 2 | 0.1250 |
| | | | 3 | 0.0525 |
| 7F | SPEP3-1 | 0.0212 | 1 | 0.0469 |
| | | | 2 | 0.1723 |
| | | | 3 | 0.2391 |
| | SPEP3-2 | 0.0285 | 1 | 0.1096 |
| | | | 2 | 0.0352 |
| | | | 3 | 0.1719 |
| | SPEP3-3 | 0.0277 | 1 | 0.1339 |
| | | | 2 | 0.1133 |
| | | | 3 | 0.0531 |

and Pearson’s correlation coefficient when evaluated on the synthetic (training) and real-world (validation and testing) problem sets. These tests show that the Ensemble-3 solutions can achieve low predictive error and a high correlation with the true PGPC performance, for both

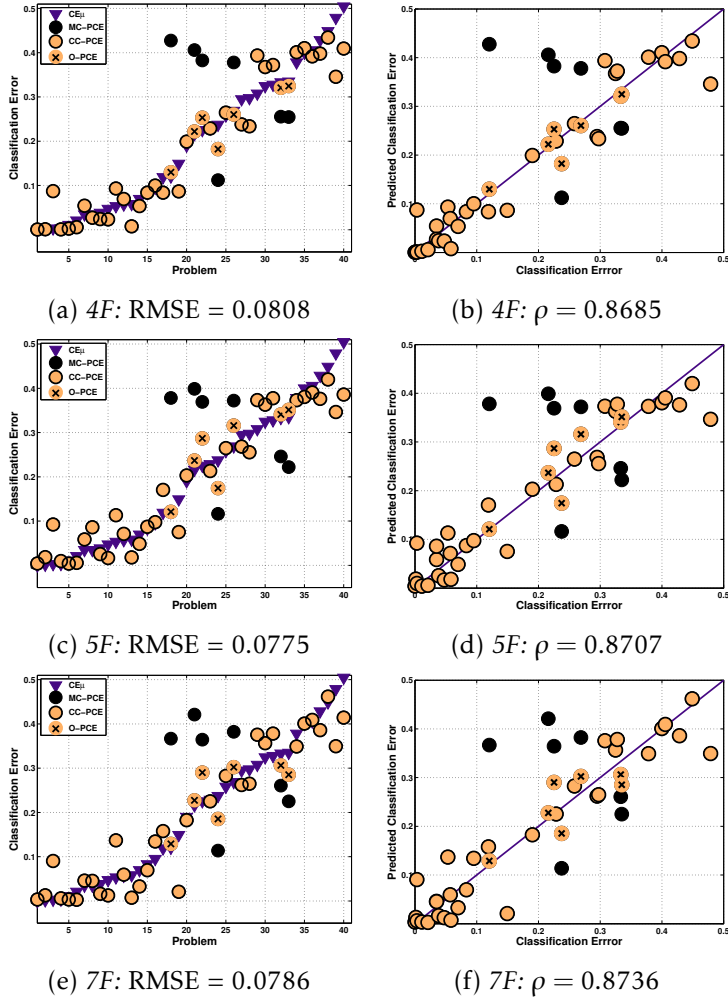


Figure 6.6: Performance prediction of the best Ensemble-3 solutions for each feature set: 4F(top), 5F(middle) and 7F (bottom).

synthetic and real-world problems. In all feature sets the correlation on synthetic problems is above 0.97, while the best performance on the real-world problems is achieved using set 5F based on RMSE and set 7F based on correlation.

SPEP: SPECIALIST PREDICTORS OF EXPECTED PERFORMANCE

Table 6.6: Performance on the SPEP selection problem for all tested classifiers, showing the median classification error from 100 independent runs. The performance is given on the training and testing sets, with bold indicating the best performance on each feature set.

| <i>Algorithm</i> | | EDC | MDC | NBC | SVM | KNN | TBC | PGPC |
|------------------|-----------------|--------|---------------|--------|--------|--------|---------------|--------|
| 4F | <i>training</i> | 0.2533 | 0.1967 | 0.0833 | 0.1067 | 0.0467 | 0.0167 | 0.0633 |
| | <i>testing</i> | 0.4722 | 0.3056 | 0.3889 | 0.3611 | 0.3056 | 0.2778 | 0.3611 |
| 5F | <i>training</i> | 0.2500 | 0.1933 | 0.0833 | 0.1033 | 0.0533 | 0.0200 | 0.0667 |
| | <i>testing</i> | 0.5000 | 0.3056 | 0.4167 | 0.3611 | 0.3333 | 0.3056 | 0.3333 |
| 7F | <i>training</i> | 0.2467 | 0.1867 | 0.0800 | 0.0533 | 0.0567 | 0.0167 | 0.0667 |
| | <i>testing</i> | 0.5000 | 0.3056 | 0.3889 | 0.4444 | 0.3333 | 0.3333 | 0.3333 |

Table 6.7: Performance on the SPEP selection problem for all tested classifiers, showing the classification error of the best solution found, evaluated over all real-world problems, with bold indicating the best performance on each feature set.

| Feature Set | EDC | MDC | NBC | SVM | KNN | TBC | PGPC |
|-------------|--------|--------|--------|--------|--------|---------------|--------|
| 4F | 0.4750 | 0.3000 | 0.4000 | 0.3500 | 0.3000 | 0.2250 | 0.2500 |
| 5F | 0.5000 | 0.3000 | 0.4250 | 0.3500 | 0.3250 | 0.1750 | 0.2500 |
| 7F | 0.5000 | 0.3000 | 0.3750 | 0.4500 | 0.3250 | 0.2500 | 0.3000 |

Focusing on the real-world problems, Figure 6.6 summarizes the performance of the Ensemble-3 predictors using each feature set (each row of the figure). These plots illustrate the performance of the achieved prediction. As in the Ensemble-2 case, it is important to highlight the impact of misclassified problems (shown as a black circle) compared to the prediction on the same problem if the correct SPEP had been chosen (O-PCE). In this case we can see more misclassifications. The reason is evident in Figure 6.5, since Group 2 and Group 3 are not so easily differentiated. However, the impact of the misclassified problems is not as large as it is for the Ensemble-2 solution, given the comparatively similar RMSE of both the Ensemble-3 and the Ensemble-2 solutions.

Table 6.8: Ensemble-3 prediction accuracy using each feature set (4F, 5F and 7F), using the best evolved SPEPs and the best classifiers with each feature set. Performance is given based on the RMSE and Pearson’s correlation coefficient when evaluated on the synthetic and real-world problem sets; with bold indicating the best performance on real-world problems.

| Feature Set | Synthetic | | Real-world | |
|-------------|---------------|---------------|---------------|---------------|
| | RMSE | correlation | RMSE | correlation |
| <i>4F</i> | 0.0288 | 0.9704 | 0.0808 | 0.8685 |
| <i>5F</i> | 0.0300 | 0.9687 | 0.0775 | 0.8707 |
| <i>7F</i> | 0.0285 | 0.9714 | 0.0786 | 0.8736 |

SPEP: SPECIALIST PREDICTORS OF EXPECTED PERFORMANCE

7

DISCUSSION

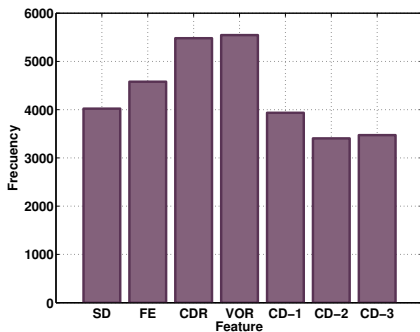
This research presents three approaches towards solving the performance prediction problem using the general PEP approach: a single PEP, an Ensemble-2 solution (2 SPEPs) and an Ensemble-3 solution (3 SPEPs). Table 7.1 presents a comparison of the best results of each solution evaluated on the real-world test cases. While all solutions achieve comparable results, it is clear that the Ensemble-2 solution achieves the lowest RMSE and the highest correlation, particularly when using set 5F. These results provide two important insights. First, that the ensemble approach is justified in this domain, with both ensembles outperforming the single PEP models. Second, that grouping the problem into useful subsets based on performance can be solved using two broad categories, what might be considered as *easy* and *difficult* problems. However, differentiating problems further becomes difficult given the underlying distribution of problems within feature space, as shown in Figure 6.5 and confirmed by the lower performance of the Ensemble-3 solution.

Starting with the relative importance of each feature used to predict performance. Since all PEPs and SPEPs were generated using symbolic regression with GP, we use statistics over the GP runs to measure the importance of each feature. Figure 7.1 shows two plots that quantify the frequency of feature use when the models were evolved using the complete feature set (7F) over 100 independent runs. Figure 7.1(a) is a bar plot where the frequency is given by summing the number of times that each feature appeared as a terminal element in the best symbolic regression solutions from each run. Figure 7.1(b) plots

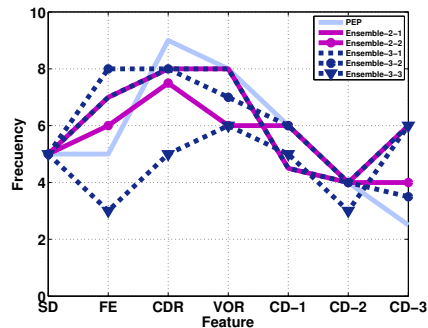
DISCUSSION

Table 7.1: A comparison of each predictor approach; where bold indicates best performance.

| | PEP | | SPEP Ensemble-2 | | SPEP Ensemble-3 | |
|----|--------|-------------|-----------------|---------------|-----------------|-------------|
| | RMSE | correlation | RMSE | correlation | RMSE | correlation |
| 4F | 0.0828 | 0.8634 | 0.0818 | 0.8717 | 0.0808 | 0.8685 |
| 5F | 0.0929 | 0.8823 | 0.0736 | 0.8981 | 0.0775 | 0.8707 |
| 7F | 0.0930 | 0.8046 | 0.0897 | 0.8514 | 0.0786 | 0.8736 |



(a) 7F



(b) 7F

Figure 7.1: Feature selection by the symbolic regression GP used to evolve all PEP and SPEP models, showing usage frequency over 100 runs: (a) bar plot of the total number of times that each feature appeared as a terminal element in the best models; and (b) median of the number of times that each feature appeared in each tree.

the median of the number of times that each feature appears in the best solution from each run. In this plot each line corresponds to either a single PEP or a particular SPEP from each ensemble; for instance, for the Ensemble-2 solutions there are two SPEPs labeled as Ensemble-2-1 and Ensemble-2-2, and similarly for the Ensemble-3 models. Notice that in this plot the lines for SPEP Ensemble-2-1 and SPEP Ensemble-3-1 overlap since they correspond to the same problem group.

Figure 7.1 reveals some interesting facts of how the symbolic regression system performs feature selection. As shown in Figure 5.11,

the features with higher correlation to PGPC performance are FE, VOR, CDR and CD-1, in that order. However, if we consider all evolved models (Figure 7.1(a)) FE is not the most widely used feature, the evolved models consistently select VOR and CDR at a higher frequency. On the other hand, the less correlated features SD, CD-2 and CD-3 are indeed used less by GP.

If we consider feature frequency in finer detail by comparing the frequency in the PEP models with the frequency in each SPEP, some interesting trends appear, as shown in Figure 6.5(b). In this case it is clear that some features are better predictors of PGPC performance on particular problem groups. For instance, CDR and VOR are the most used by the PEP models. On the other hand, FE is used with a higher frequency when predicting performance on easier problems (Ensemble-2-1, Ensemble-3-1) than for the hardest problems (Ensemble-3-3). This is also the case for CDR and slightly for CD-2. Conversely, while CD-3 is rarely used in PEP models, it appears to be very useful in predicting performance on the most difficult problems (Ensemble-3-3) and the easiest (Ensemble-2-1 and Ensemble-3-1) problems.

It is also instructive to determine if the dimensionality reduction applied as preprocessing has a negative effect with regards to performance prediction. Our proposal is to use the first two principal components of the data, in order to simplify the description of the real-world problems. However, it is not clear if the percentage of the variance described by such few components is enough to properly characterize the problems. To analyze this, Figure 7.2 presents scatter plots of all the real-world problems $p \in Q'$, showing the percentage of the total variance of the data explained by the first two principal components (x-axis) and the prediction error (PE) (y-axis) computed as the absolute difference between CE_{μ} and PCE. In particular, Figure 7.2(a) is based on the PEP-4F model while Figure 7.2(b) is based on the SPEP-2-5F model. The legend of each plot specifies the computed Pearson's correlation coefficient ρ between both measures. Notice that there is no significant correlation, suggesting that the accuracy of the models does not suffer from the proposed preprocessing.

DISCUSSION

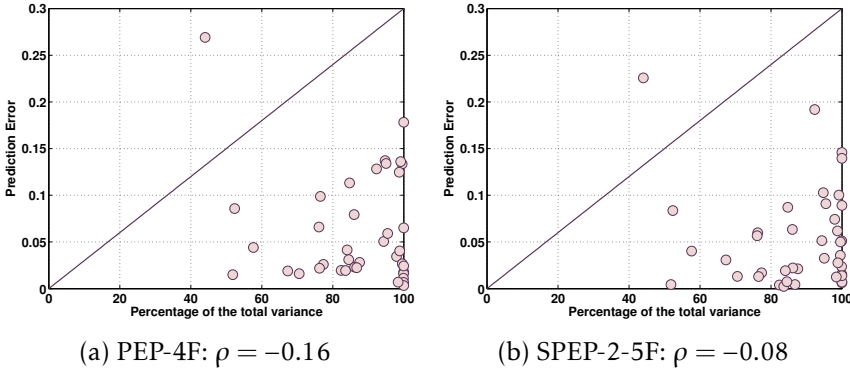


Figure 7.2: Scatter plots show the relationship between the percentage of the total variance explained by two principal components (x-axis) and the prediction error (y-axis), for all problems $p \in Q'$, where the prediction error is the absolute difference between the $CE\mu$ and PCE, figure on the left show the PEP-4F model and figure on the right SPEP-2-5F, where ρ specifies Pearson's correlation coefficient.

Finally, an implicit goal of the PEP and SPEP models is to obtain accurate performance predictions in a fraction of the time required to obtain those same estimates by actually performing the GP runs. Pragmatically, one way to validate if this goal is achieved is to calculate the running time for all problems, based on the employed PGPC implementation and the complete SPEP process. These experiments were conducted using MATLAB and the GPLAB toolbox (Silva and Almeida, 2003) running on a PC with Ubuntu 12.04 LTS using an Intel Xeon(R) CPU E3-1270 v3 @ 3.50GHz x 8 processor with 15.6 GB of RAM. In these tests, the minimum amount of time required to compute $CE\mu$ (30 runs of PGPC) was 3360.96 seconds, while the maximum amount of time required to compute the PCE (running the SPEP process) was 11.22 seconds. These results clearly show that PEP and SPEP models can be used in real-world scenarios to obtain both accurate and efficient estimations of GP performance ¹.

¹It is important to state that our PGPC and SPEP implementations were not implemented in any optimal way and that running times might be different.

8

CONCLUSIONS

This work presents three main contributions. First, extensions of the PEP approach originally proposed in Trujillo et al. (2011b,a,c), by adding new descriptive measures and testing the PEP models built with synthetic classification problems over a more challenging scenario, performance prediction on real-world classification problems with different dimensions and class imbalance. To achieve the latter we included a preprocessing step for dimensionally reduction, something that previous proposals lacked. Second, the proposed models predict the performance of the GP classifier when they are evaluated on the test set of fitness cases, while previous works focused on predicting training performance. For real-world scenarios, predicting the test performance of a learning algorithm is more relevant since overfitting can appear on difficult problem instances. Third, this work presents a new proposal using an ensemble of SPEPs, where the problems are separated into groups and specialized models were built for each group, improving the prediction accuracy on unseen real-world problems.

The main conclusions derived from this work are the following. First, the proposed dimensionality reduction was successful, it allowed the system to learn the predictive models using simple 2D synthetic problems and apply them on real-world problems with considerably more dimensions. Second, the evolved PEP and SPEP models were able to accurately predict PGPC performance on imbalanced datasets, without the need of using imbalanced data during the training phase. Third, the new descriptive measures proposed in this work (CD-1, CD-2, CD-3) complemented the problem descriptors used in previous works to

CONCLUSIONS

help improve predictive accuracy. Some of the proposed descriptors (CD-1) were among the most correlated with PGPC performance; their usefulness was confirmed when analyzing the feature selection performed by GP. However, it's important to note that all descriptors were used in most evolved PEPs, even if some descriptors exhibited very small amounts of correlation with PGPC performance. Finally, our ensemble proposal provides two general perspectives of the prediction task: categorical and numerical prediction. Where, a categorical prediction is used to select specific SPEPs, while the numerical prediction is given by the chosen SPEP. While not explored in this work, the categorical prediction might be sufficient for some applications, such as in fuzzy inference systems.

Finally, possible future work derived from this research includes the following. The problem descriptors used in this work produced good results, but defining the optimal set of descriptors is still an open question. We will also use this methodology for many classifiers, deriving one PEP for each classifier, thus allowing us to create an expert system for classifier selection. Another possibility is to use the PEPs within a wrapper approach, where the PEP model could be used as a surrogate fitness function for GP-based classifiers. Moreover, these methodologies could be extended to predict the performance of a GP-based symbolic regression system, building PEP models using a set of descriptive measures that can characterize symbolic regression problems accurately. To do so, a proper dimensionality reduction step must be developed.

- Altenberg, L. (1994). *The evolution of evolvability in genetic programming*, pages 47–74. MIT Press, Cambridge, MA, USA.
- Altenberg, L. (1997). Fitness distance correlation analysis: An instructive counterexample. In *In Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 57–64. Morgan Kaufmann.
- Anderson, T. W. (1958). *Introduction to multivariate statistical analysis*. John Wiley & Sons, Inc., New York, NY, USA.
- Angeline, P. J. (1997). Subtree crossover: Building block engine or macromutation? In Koza, J. R., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M., Iba, H., and Riolo, R. L., editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 9–17, Stanford University, CA, USA. Morgan Kaufmann.
- Bentley, P. J. (2000). “Evolutionary, my dear Watson” Investigating Committee-based Evolution of Fuzzy Rules for the Detection of Suspicious Insurance Claims. In *Genetic and Evolutionary Computation Conf.(GECCO-2000)*, pages 702–709.
- Clergue, M., Collard, P., Tomassini, M., and Vanneschi, L. (2002). Fitness distance correlation and problem difficulty for genetic programming. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, USA, 9-13 July 2002*, pages 724–732.
- Derrac, J., García, S., Molina, D., and Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18.

- Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification (2Nd Edition)*. Wiley-Interscience.
- Eggermont, J., Kok, J. N., and Kusters, W. A. (2004). Genetic programming for data classification: partitioning the search space. In *Proceedings of the 2004 ACM symposium on Applied computing, SAC '04*, pages 1001–1005, New York, NY, USA. ACM.
- Eiben, A. E. and Smith, J. E. (2003). *Introduction to Evolutionary Computing*. SpringerVerlag.
- Folino, G., Pizzuti, C., and Spezzano, G. (2010). An ensemble-based evolutionary framework for coping with distributed intrusion detection. *Genetic Programming and Evolvable Machines*, 11(2):131–146.
- Friedman, M. (1937). The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, 32(200):675–701.
- Galván-López, E., Dignum, S., and Poli, R. (2008). The effects of constant neutrality on performance and problem hardness in gp. In *Proceedings of the 11th European conference on Genetic programming, EuroGP'08*, pages 312–324, Berlin, Heidelberg. Springer-Verlag.
- Galván-López, E., McDermott, J., O'Neill, M., and Brabazon, A. (2010). Defining locality in genetic programming to predict performance. In *IEEE Congress on Evolutionary Computation*, pages 1–8.
- Galván-López, E., McDermott, J., O'Neill, M., and Brabazon, A. (2011). Defining locality as a problem difficulty measure in genetic programming. *Genetic Programming and Evolvable Machines*, 12(4):365–401.
- Galván-López, E. and Poli, R. (2006a). An empirical investigation of how and why neutrality affects evolutionary search. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO '06*, pages 1149–1156, New York, NY, USA. ACM.
- Galván-López, E. and Poli, R. (2006b). Some steps towards understanding how neutrality affects evolutionary search. In Runarsson,

- T., Beyer, H.-G., Burke, E., Merelo-Guervós, J., Whitley, L., and Yao, X., editors, *Parallel Problem Solving from Nature - PPSN IX*, volume 4193 of *Lecture Notes in Computer Science*, pages 778–787. Springer Berlin Heidelberg.
- Gathercole, C. and Ross, P. (1994a). Dynamic training subset selection for supervised learning in genetic programming. In *Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature*, PPSN III, pages 312–321, London, UK, UK. Springer-Verlag.
- Gathercole, C. and Ross, P. (1994b). Some training subset selection methods for supervised learning in genetic programming, presented at ecai'94 workshop on applied genetic and other evolutionary algorithms.
- Goldberg, D. E. (1987). Simple genetic algorithms and the minimal, deceptive problem. In Davis, L., editor, *Genetic algorithms and simulated annealing*, Research Notes in Artificial Intelligence, pages 74–88. Pitman, London.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.
- Gonçalves, I. and Silva, S. (2011). Experiments on controlling overfitting in genetic programming. In *Proceedings in Artificial Intelligence, 15th Portuguese Conference on Artificial Intelligence, EPIA 2011, October 10-13.*, Lecture Notes in Computer Science. Springer.
- Gonçalves, I. and Silva, S. (2013). Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In Krawiec, K., Moraglio, A., Hu, T., Etaner-Uyar, A., and Hu, B., editors, *Genetic Programming*, volume 7831 of *Lecture Notes in Computer Science*, pages 73–84. Springer Berlin Heidelberg.

- Graff, M., Escalante, H. J., Cerda-Jacobo, J., and Gonzalez, A. A. (2013a). Models of performance of time series forecasters. *Neurocomputing*, 122(0):375 – 385. Advances in cognitive and ubiquitous computing Selected papers from the Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2012).
- Graff, M. and Poli, R. (2008). Practical model of genetic programming’s performance on rational symbolic regression problems. In *EuroGP*, pages 122–133.
- Graff, M. and Poli, R. (2010). Practical performance models of algorithms in evolutionary program induction and other domains. *Artif. Intell.*, 174(15):1254–1276.
- Graff, M. and Poli, R. (2011). Performance models for evolutionary program induction algorithms based on problem difficulty indicators. In *Proceedings of the 14th European conference on Genetic programming, EuroGP’11*, pages 118–129, Berlin, Heidelberg. Springer-Verlag.
- Graff, M., Poli, R., and Flores, J. J. (2013b). Models of performance of evolutionary program induction algorithms based on indicators of problem difficulty. *Evolutionary Computation*, 21(4):533–560.
- Guo, H., Jack, L., and Nandi, A. (2005). Feature generation using genetic programming with application to fault classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 35(1):89–99.
- He, J., Chen, T., and Yao, X. (2015). On the easiest and hardest fitness functions. *Evolutionary Computation, IEEE Transactions on*, 19(2):295–305.
- Hengpraprom, S. and Chongstitvatana, P. (2008). A genetic programming ensemble approach to cancer microarray data classification. In *Innovative Computing Information and Control, 2008. ICICIC ’08. 3rd International Conference on*, pages 340–340.

- Ho, T. K. and Basu, M. (2002). Complexity measures of supervised classification problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(3):289–300.
- Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70.
- Imamura, K., Soule, T., Heckendorn, R., and Foster, J. (2003). Behavioral diversity and a probabilistically optimal gp ensemble. *Genetic Programming and Evolvable Machines*, 4(3):235–253.
- Jones, T. and Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Kauffman, S. and Levin, S. (1987). Towards a general theory of adaptive walks on rugged landscapes. *Journal of Theoretical Biology*, 128(1):11–45.
- Kimura, M. (1983). *The neutral theory of molecular evolution*. Cambridge University Press.
- Kinnear, K. E. (1994). Fitness landscapes and difficulty in genetic programming. In *Proceedings of the First IEEE Conference on Evolutionary Computing*, pages 142–147, Piscataway, NY. IEEE Press.
- Kotsiantis, S. B., Zaharakis, I. D., and Pintelas, P. E. (2006). Machine learning: A review of classification and combining techniques. *Artif. Intell. Rev.*, 26(3):159–190.
- Koza, J. (2010). Human-competitive results produced by genetic programming. *Genetic Programming and Evolvable Machines*, 11(3):251–284.
- Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA.

- Koza, J. R. (1994). *Genetic programming II: automatic discovery of reusable programs*. MIT Press, Cambridge, MA, USA.
- Langdon, W. B. and Poli, R. (2002). *Foundations of genetic programming*. Springer.
- Lasarczyk, C. W. G., Dittrich, P. W. G., and Banzhaf, W. W. G. (2004). Dynamic subset selection based on a fitness case topology. *Evol. Comput.*, 12(2):223–242.
- Lichman, M. (2013). UCI machine learning repository.
- Malan, K. and Engelbrecht, A. P. (2014). Particle swarm optimisation failure prediction based on fitness landscape characteristics. In *2014 IEEE Symposium on Swarm Intelligence, SIS 2014, Orlando, FL, USA, December 9-12, 2014*, pages 149–157.
- Malan, K. M. and Engelbrecht, A. P. (2013). A survey of techniques for characterising fitness landscapes and some possible ways forward. *Inf. Sci.*, 241:148–163.
- Martínez, Y., Trujillo, L., Naredo, E., and Legrand, P. (2014). A comparison of fitness-case sampling methods for symbolic regression with genetic programming. In Tantar, A.-A. et al., editors, *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V*, volume 288 of *Advances in Intelligent Systems and Computing*, pages 201–212. Springer International Publishing.
- Martínez, Y., Naredo, E., Trujillo, L., and López, E. G. (2013). Searching for novel regression functions. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2013, Cancun, Mexico, June 20-23, 2013*, pages 16–23.
- Martínez, Y., Trujillo, L., Galván-López, E., and Legrand, P. (2012). A comparison of predictive measures of problem difficulty for classification with genetic programming. In *ERA 2012*, Tijuana, Mexico.
- McClymont, K., Walker, D., and Dupenois, M. (2012). The lay of the land: a brief survey of problem understanding. In *Proceedings of the*

- fourteenth international conference on Genetic and evolutionary computation conference companion*, GECCO Companion '12, pages 425–432, New York, NY, USA. ACM.
- McDermott, J., White, D. R., Luke, S., Manzoni, L., Castelli, M., Vanneschi, L., Jaskowski, W., Krawiec, K., Harper, R., De Jong, K., and O'Reilly, U.-M. (2012). Genetic programming needs better benchmarks. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, GECCO '12, pages 791–798, New York, NY, USA. ACM.
- McPhee, N., Ohs, B., and Hutchison, T. (2008). Semantic building blocks in genetic programming. In O'Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A., De Falco, I., Della Cioppa, A., and Tarantino, E., editors, *Genetic Programming*, volume 4971 of *Lecture Notes in Computer Science*, pages 134–145. Springer Berlin Heidelberg.
- Michie, D., Spiegelhalter, D. J., Taylor, C. C., and Campbell, J., editors (1994). *Machine learning, neural and statistical classification*. Ellis Horwood, Upper Saddle River, NJ, USA.
- Mitchell, T. M. (1980). The need for biases in learning generalizations. Technical report.
- Moraglio, A., Krawiec, K., and Johnson, C. G. (2012). Geometric semantic genetic programming. In *Parallel Problem Solving from Nature - PPSN XII - 12th International Conference, Taormina, Italy, September 1-5, 2012, Proceedings, Part I*, pages 21–31.
- Muharram, M. and Smith, G. (2005). Evolutionary constructive induction. *Knowledge and Data Engineering, IEEE Transactions on*, 17(11):1518–1528.
- Muñoz, L., Silva, S., and Trujillo, L. (2015). M3GP - multiclass classification with GP. In *Genetic Programming - 18th European Conference, EuroGP 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings*, pages 78–91.

- O'Neill, M., Vanneschi, L., Gustafson, S., and Banzhaf, W. (2010). Open issues in genetic programming. *Genetic Programming and Evolvable Machines*, 11(3-4):339–363.
- Poli, R. and Galván-López, E. (2012). The effects of constant and bit-wise neutrality on problem hardness, fitness distance correlation and phenotypic mutation rates. *Evolutionary Computation, IEEE Transactions on*, 16(2):279–300.
- Poli, R., Graff, M., and McPhee, N. F. (2009). Free lunches for function and program induction. In *Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms, FOGA '09*, pages 183–194, New York, NY, USA. ACM.
- Poli, R. and McPhee, N. F. (2008). Parsimony pressure made easy. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1267–1274, New York, NY, USA. ACM.
- Punch, B., Zongker, D., and Goodman, E. (1996). Advances in genetic programming. chapter The Royal Tree Problem, a Benchmark for Single and Multiple Population Genetic Programming, pages 299–316. MIT Press, Cambridge, MA, USA.
- Qing-Shan, C., gg De-fu, Li-Jun, W., and Huo-Wang, C. (2007). A modified genetic programming for behavior scoring problem. In *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on*, pages 535–539.
- Quick, R., Rayward-Smith, V., and Smith, G. (1998). Fitness distance correlation and ridge functions. In Eiben, A., Bäck, T., Schoenauer, M., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature — PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 77–86. Springer Berlin Heidelberg.
- Rothlauf, F. (2006). *Representations for Genetic and Evolutionary Algorithms*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

- Samuel, A. L. (1983). Ai: Where it has been and where it is going. In *Proc. of the 8th IJCAI*, pages 1152–1157, Karlsruhe, Germany.
- Schaffer, C. (1994). A conservation law for generalization performance. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 259–265. Morgan Kaufmann.
- Sherrah, J. R., Bogner, R. E., and Bouzerdoum, A. (1997). 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.
- Silva, S. and Almeida, J. (2003). GPLAB - A Genetic Programming Toolbox for MATLAB. In *Gregersen L (ed), Proceedings of the Nordic MATLAB Conference*, pages 273—278.
- Silva, S. and Costa, E. (2009). Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. *Genetic Programming and Evolvable Machines*, 10(2):141–179.
- Smith, M. and Bull, L. (2005). Genetic programming with a genetic algorithm for feature construction and selection. *Genetic Programming and Evolvable Machines*, 6(3):265–281.
- Sohn, S. Y. (1999). Meta analysis of classification algorithms for pattern recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(11):1137–1144.
- Sotelo, A., Guijarro, E., Coria, L. N., Trujillo, L., and Valle, P. A. (2012). Epilepsy ictal stage identification by 0-1 test of chaos. In *Proceedings of the third IFAC CHAOS Conference, CHAOS 2012*, pages 223–228. IFAC.
- Sotelo, A., Guijarro, E., García, M., and Vázquez, C. (2007). Epoch parameterization by gabor atom density in experimental epilepsy. In *4th International Conference on Electrical and Electronics Engineering*, pages 61–64.

- Sotelo, A., Guijarro, E., García, M., and Vázquez, C. (2008). Epilepsy stages diagnosis by gabor atom density according to their aspect ratio. In *4th International Conference on Electrical Engineering, Computing Science and Automatic Control*, pages 158–163.
- Sotelo, A., Guijarro, E., Trujillo, L., Coria, L. N., and Martínez, Y. (2013). Identification of epilepsy stages from ecog using genetic programming classifiers. *Comp. in Bio. and Med.*, 43(11):1713–1723.
- Spector, L. (2012). Assessment of problem modality by differential performance of lexibase selection in genetic programming: a preliminary report. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion*, GECCO Companion '12, pages 401–408. ACM.
- Stadler, P. (2002). Fitness landscapes. In Lässig, M. and Valleriani, A., editors, *Biological Evolution and Statistical Physics*, volume 585 of *Lecture Notes in Physics*, pages 183–204. Springer Berlin Heidelberg.
- Tanigawa, T. and Zhao, Q. (2000). A study on efficient generation of decision trees using genetic programming. In *Proc. Genetic and Evolutionary Computation Conference (GECCO'2000), Las Vegas*, pages 1047–1052. Morgan Kaufmann.
- Tomassini, M., Vanneschi, L., Collard, P., and Clergue, M. (2005). A study of fitness distance correlation as a difficulty measure in genetic programming. *Evol. Comput.*, 13(2):213–239.
- Trawinski, B., Smetek, M., Telec, Z., and Lasota, T. (2012). Nonparametric statistical analysis for multiple comparison of machine learning regression algorithms. *Applied Mathematics and Computer Science*, 22(4):867–881.
- Trujillo, L., Martínez, Y., Galván-López, E., and Legrand, P. (2011a). Predicting problem difficulty for genetic programming applied to data classification. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 1355–1362, New York, NY, USA. ACM.

- Trujillo, L., Martínez, Y., López, E. G., and Legrand, P. (2012). A comparative study of an evolvability indicator and a predictor of expected performance for genetic programming. In *Genetic and Evolutionary Computation Conference, GECCO '12, Philadelphia, PA, USA, July 7-11, 2012, Companion Material Proceedings*, pages 1489–1490.
- Trujillo, L., Martínez, Y., and Melin, P. (2011b). Estimating classifier performance with genetic programming. In *Proceedings of the 14th European conference on Genetic programming, EuroGP'11*, pages 274–285, Berlin, Heidelberg. Springer-Verlag.
- Trujillo, L., Martínez, Y., and Melin, P. (2011c). How many neurons?: A genetic programming answer. In *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO '11*, pages 175–176, New York, NY, USA. ACM.
- Trujillo, L., Spector, L., Naredo, E., and Martínez, Y. (2013). A behavior-based analysis of modal problems. In *Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013, Companion Material Proceedings*, pages 1047–1054.
- Tsakonas, A. (2006). A comparison of classification accuracy of four genetic programming-evolved intelligent structures. *Information Sciences*, 176(6):691 – 724.
- Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 49:433–460.
- Uy, N. Q., Hoai, N. X., O'Neill, M., McKay, R. I., and Galván-López, E. (2011). Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119.
- Vanneschi, L., Castelli, M., and Manzoni, L. (2011). The k landscapes: A tunably difficult benchmark for genetic programming. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO '11*, pages 1467–1474, New York, NY, USA. ACM.

- Vanneschi, L., Castelli, M., and Silva, S. (2014). A survey of semantic methods in genetic programming. *Genetic Programming and Evolvable Machines*, 15(2):195–214.
- Vanneschi, L., Clergue, M., Collard, P., Tomassini, M., and Verel, S. (2004). Fitness clouds and problem hardness in genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'04*, pages 690–701.
- Vanneschi, L., Tomassini, M., Collard, P., and Clergue, M. (2003). Fitness distance correlation in genetic programming: a constructive counterexample. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2003, 8 - 12 December 2003, Canberra, Australia*, pages 289–296.
- Vanneschi, L., Tomassini, M., Collard, P., and Verel, S. (2006). Negative slope coefficient: A measure to characterize genetic programming fitness landscapes. In *Genetic Programming, 9th European Conference, EuroGP 2006, Budapest, Hungary, April 10-12, 2006, Proceedings*, pages 178–189.
- Vanneschi, L., Tomassini, M., Collard, P., Vérel, S., Pirola, Y., and Mauri, G. (2007). A comprehensive view of fitness landscapes with neutrality and fitness clouds. In *Proceedings of the 10th European conference on Genetic programming, EuroGP'07*, pages 241–250, Berlin, Heidelberg. Springer-Verlag.
- Vanneschi, L., Valsecchi, A., and Poli, R. (2009). Limitations of the fitness-proportional negative slope coefficient as a difficulty measure. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, GECCO '09*, pages 1877–1878, New York, NY, USA. ACM.
- Verel, S., Collard, P., and Clergue, M. (2003). Where are bottlenecks in NK fitness landscapes? In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2003, 8 - 12 December 2003, Canberra, Australia*, pages 273–280.

- Wolpert, D. and Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.
- Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural Comput.*, 8(7):1341–1390.
- Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proceedings of the Sixth International Congress of Genetics*, 1:356–66.
- Yu, T. and Miller, J. (2001). Neutrality and the evolvability of boolean function landscape. In Miller, J., Tomassini, M., Lanzi, P., Ryan, C., Tettamanzi, A., and Langdon, W., editors, *Genetic Programming*, volume 2038 of *Lecture Notes in Computer Science*, pages 204–217. Springer Berlin Heidelberg.
- Z-Flores, E., Trujillo, L., Schütze, O., and Legrand, P. (2015). A local search approach to genetic programming for binary classification. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference, GECCO '15*, pages 1151–1158, New York, NY, USA. ACM.
- Zhang, M. and Smart, W. (2004). Multiclass object classification using genetic programming. In Raidl, G., Cagnoni, S., Branke, J., Corne, D., Drechsler, R., Jin, Y., Johnson, C., Machado, P., Marchiori, E., Rothlauf, F., Smith, G., and Squillero, G., editors, *Applications of Evolutionary Computing*, volume 3005 of *Lecture Notes in Computer Science*, pages 369–378. Springer Berlin Heidelberg.
- Zhang, M. and Smart, W. (2006). Using gaussian distribution to construct fitness functions in genetic programming for multiclass object classification. *Pattern Recogn. Lett.*, 27(11):1266–1274.
- Zhou, Z.-H. (2012). *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC, 1st edition.

BIBLIOGRAPHY



NO FREE LUNCH THEOREMS (NFL)

Broadly speaking, there are two no free lunch theorems. One for supervised machine learning Wolpert (1996) and one for search/optimization (Wolpert and Macready, 1997). For an overview of the (no) free lunch and associated theorems, see David Wolpert's What does dinner cost?

A.1 NO FREE LUNCH FOR SUPERVISED MACHINE LEARNING

Hume (1739–1740) pointed out that even after the observation of the frequent or constant conjunction of objects, we have no reason to draw any inference concerning any object beyond those of which we have had experience. More recently, and with increasing rigour, Mitchell (1980), Schaffer (1994) and Wolpert (1996) showed that bias-free learning is futile.

Wolpert (1996) shows that in a noise-free scenario where the loss function is the misclassification rate, if one is interested in off-training-set error, then there are no a priori distinctions between learning algorithms.

More formally, where

\mathbb{T} = training set;

I = number of elements in training set;

ν = target input-output relationships;

h = hypothesis (the algorithm's guess for ν made in response to \mathbb{T});

\mathcal{L} = off-training-set loss associated with ν and h (generalization error)

all algorithms are equivalent, on average, by any of the following measures of risk: $E(\mathcal{L} | \mathbb{T})$, $E(\mathcal{L} | I)$, $E(\mathcal{L} | \nu, \mathbb{T})$ or $E(\mathcal{L} | \nu, I)$.

How well you do is determined by how aligned your learning algorithm $P(h | \mathbb{T})$ is with the actual posterior, $P(\nu | \mathbb{T})$.

Wolpert's result, in essence, formalizes Hume, extends him and calls the whole of science into question.

A.2 NO FREE LUNCH FOR SEARCH/OPTIMIZATION

The no free lunch theorem for search and optimization (Wolpert and Macready, 1997) applies to finite spaces and algorithms that do not resample points. All algorithms that search for an extremum of a cost function perform exactly the same when averaged over all possible cost functions. So, for any search/optimization algorithm, any elevated performance over one class of problems is exactly paid for in performance over another class.