

# Evolutionary Induction of Projection Maps for Feature Extraction

THESIS SUBMITTED

IN ACCORDANCE WITH THE REQUIREMENTS OF

THE UNIVERSITY OF LIVERPOOL

FOR THE DEGREE OF

DOCTOR IN PHILOSOPHY

By

Eduardo Rodriguez Martinez

Department of Electrical Engineering and Electronics

The University of Liverpool

Liverpool, United Kingdom

2011

*Dedicated to my parents Socorro and Eduardo,  
and to my brother Omar.*

# Abstract

This thesis proposes an evolutionary scheme for automatic design of feature extraction methods, tailored to a given classification problem. The main advantage of the proposed scheme is its capacity to formulate new models when the existing ones do not fit the problem at hand. The learning phase is expressed as a model selection problem, where the best performing model is selected among the genetic pool, assessed by an estimation of out-of-sample generalization error. Each individual in the genetic pool represents a potential model encoded into a hybrid genotype, specifically designed to hold a tree structure and an scalar array to represent both feature-extraction and classification stages. The role of the inducer is to automatically design a mapping function to be used as the core of the feature-extraction stage, as well as fine-tune the corresponding hyper-parameters for the feature-extraction/classification pair. Two paradigms are explored to express the feature-extraction stage, namely projection pursuit and spectral embedding methods.

Both paradigms can express several feature extraction algorithms under a common template. In the case of projection pursuit, such template consist on the optimisation of a cost function, also known as projection index, that can be specifically designed to highlight certain properties of the extracted features. While for spectral embedding methods, a suitable set of similarity metrics is needed to construct a weight matrix, which encodes the links between any two samples on the vertices of a graph. The eigendecomposition of such weight matrix represents the solution to an optimisation problem looking for a low-dimensional space, retaining the characteristics described by the original distance metric. The proposed inducer evolves an optimal projection index or a desired distance metric for the corresponding feature-extraction paradigm. Additionally, projection pursuit was extended to the nonlinear case by means of the kernel trick. The determination of a nonlinear residual subspace for sequential projection pursuit is reduced to the computation of an updated kernel matrix.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Contents</b>	<b>ii</b>
<b>List of Figures</b>	<b>v</b>
<b>List of Tables</b>	<b>vi</b>
<b>Acknowledgements</b>	<b>vii</b>
<b>Acronyms</b>	<b>viii</b>
<b>Mathematical Notation</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation and Objectives . . . . .	1
1.2 Scope and Contributions . . . . .	3
1.3 Published/Submitted Articles . . . . .	4
<b>2 Machine Learning Methodologies</b>	<b>6</b>
2.1 Types of Algorithms . . . . .	6
2.1.1 Supervised Learning . . . . .	7
2.1.2 Unsupervised Learning . . . . .	7
2.1.3 Reinforcement Learning . . . . .	7
2.1.4 Transduction . . . . .	7
2.2 Pattern Classification . . . . .	8
2.2.1 Linear Classifiers . . . . .	8
2.2.1.1 Generative classifiers . . . . .	9
2.2.1.2 Discriminative classifiers . . . . .	10
2.2.2 Nonlinear Classifiers . . . . .	12
2.2.2.1 Generative classifiers . . . . .	12
2.2.2.2 Discriminative classifiers . . . . .	13
2.3 Feature Extraction . . . . .	14

2.3.1	Projection Methods . . . . .	14
2.3.1.1	Principal component analysis . . . . .	14
2.3.1.2	Linear Fisher discriminant . . . . .	15
2.3.1.3	Independent component analysis . . . . .	16
2.3.2	Projection Methods in Kernel-Induced Feature Space . . . . .	18
2.3.2.1	Kernel principal component analysis . . . . .	18
2.3.2.2	Kernel Fisher discriminant . . . . .	19
2.3.3	Projection Pursuit . . . . .	20
2.3.3.1	The optimisation problem . . . . .	20
2.3.3.2	Current projection indices . . . . .	20
2.3.3.3	Extension to multiple dimensions . . . . .	23
2.3.4	Spectral Embedding Methods . . . . .	24
2.3.4.1	Unsupervised methods . . . . .	25
2.3.4.2	Supervised methods . . . . .	26
2.3.5	Feature Extraction for High-Dimensional Spaces . . . . .	28
<b>3</b>	<b>Optimisation Techniques in Machine Learning</b>	<b>30</b>
3.1	Standard Optimisation Techniques . . . . .	31
3.1.1	Linear Programming . . . . .	31
3.1.2	Quadratic Programming . . . . .	31
3.1.3	Nonlinear Unconstrained Programming . . . . .	33
3.2	Evolutionary Optimisation Techniques . . . . .	36
3.2.1	Genetic Algorithms . . . . .	37
3.2.2	Genetic Programming . . . . .	39
3.2.3	Particle Swarm Optimisation . . . . .	42
3.3	Evolutionary Optimisation in Pattern Recognition . . . . .	44
<b>4</b>	<b>Automated Induction of Projection Pursuit Indices</b>	<b>48</b>
4.1	Introduction . . . . .	48
4.2	Proposed Evolutionary Learning System . . . . .	49
4.2.1	Fitness Function . . . . .	50
4.2.2	Index Optimisation . . . . .	51
4.2.3	Dimensionality Control . . . . .	52
4.3	Evolutionary Framework Language Definition . . . . .	53
4.4	Experimental Results . . . . .	57
4.4.1	Datasets . . . . .	57
4.4.2	Evolution Process . . . . .	57
4.4.3	Training and Testing Set Design . . . . .	60
4.4.4	Baseline Projection Pursuit Comparison . . . . .	60
4.4.5	Comparison of EPP with Collaborative Methods . . . . .	66

4.4.6	Evolved Indices as a Tree Representation . . . . .	68
4.5	Summary . . . . .	70
<b>5</b>	<b>Nonlinear Projection Pursuit via Kernel-Induced Spaces</b>	<b>72</b>
5.1	Introduction . . . . .	72
5.2	Problem Formulation in Kernel-Induced Spaces . . . . .	73
5.3	Nonlinear Sequential Removal Process . . . . .	74
5.4	Whitening in Feature Space . . . . .	75
5.5	Experimental Results . . . . .	77
5.5.1	Datasets . . . . .	77
5.5.2	Experimental Setup . . . . .	79
5.5.3	Experimental Results and Analysis . . . . .	80
5.6	Summary . . . . .	86
<b>6</b>	<b>Evolutionary Induction of Heterogeneous Proximities for Supervised Embeddings</b>	<b>87</b>
6.1	Model Definition . . . . .	88
6.2	Evolutionary Induction . . . . .	94
6.2.1	Chromosome Encoding . . . . .	94
6.2.2	Fitness Function . . . . .	96
6.2.3	Function and Terminals . . . . .	97
6.3	Experimental Results . . . . .	100
6.3.1	Datasets . . . . .	100
6.3.2	Experimental Setup . . . . .	101
6.3.3	Experimental Results and Analysis . . . . .	102
6.4	Summary . . . . .	106
<b>7</b>	<b>Conclusions and Future Work</b>	<b>107</b>
7.1	Conclusions . . . . .	107
7.2	Future Work . . . . .	109
	<b>Bibliography</b>	<b>131</b>

# List of Figures

2.1	Artificial neural network represented as weighted graph. . . . .	13
2.2	Nonlinear projection . . . . .	18
3.1	Main genetic algorithm loop . . . . .	37
3.2	Genetic algorithm flowchart . . . . .	40
3.3	Basic subtree genetic operators for genetic programming . . . . .	42
3.4	Michigan and Pittsburgh architectures . . . . .	45
4.1	Evolutionary model selection . . . . .	49
4.2	GP fitness evaluation . . . . .	51
4.3	Possible evolved trees . . . . .	56
4.4	Induction process for wine dataset . . . . .	58
4.5	Minimisation of fitness function . . . . .	60
4.6	2D scatter plots of the best two projected features for wine data . . . .	62
4.7	2D scatter plots of the best two projected features for heart data . . . .	62
4.8	2D scatter plots of the best two projected features for cancer data . . . .	63
4.9	Synthetic dataset analysis . . . . .	65
4.10	Examples of ROC curves using the best projection methods for heart dataset. . . . .	65
4.11	Best evolved trees for different datasets . . . . .	69
5.1	Sensitivity analysis for dominant features extracted with EKPP . . . . .	83
5.2	Evolved trees for different datasets . . . . .	84
6.1	An example of a possible SSE configuration . . . . .	88
6.2	Dollar sign and swiss roll in 3D . . . . .	90
6.3	Example of proximity matrices and embeddings . . . . .	91

# List of Tables

2.1	PDF-Based Indices. . . . .	21
2.2	Moment-Based Indices. . . . .	21
2.3	Class-Information-Based Indices. . . . .	22
4.1	Language Definition. . . . .	54
4.2	Datasets Summary. . . . .	57
4.3	Comparison of the evolved indices against $\mathfrak{S}_1 - \mathfrak{S}_{10}$ . . . . .	59
4.4	Summary of Table 4.3 . . . . .	61
4.5	Contrasting collaborating methods . . . . .	67
5.1	Datasets summary: Testing, training and validation set sizes . . . . .	78
5.2	Comparison between EKPP and relevant kernel methods . . . . .	81
5.3	Computational time comparison of EKPP . . . . .	82
6.1	Classic similarity metrics in matrix notation . . . . .	95
6.2	Function set . . . . .	98
6.3	Terminal set . . . . .	99
6.4	Selected SSE methods . . . . .	100
6.5	Types definition . . . . .	101
6.6	Dataset summary . . . . .	101
6.7	Comparison of the proposed framework including median values of the classification error assessed with 10-CV, and optimal parameters for selected SSE methods consisting of size of the local neighbourhood $k$ , number of extracted features $b$ , and penalty parameter $\beta$ . . . . .	103
6.8	Best evolved models for each classification problem, along with its tree representation. . . . .	104



# Acknowledgements

My research could not have been possible without the advice and support of many people. To begin with, I want to deeply thank my supervisor Yannis Goulermas for his unlimited patience and advice. Similarly I want to thank Tingting Mu for her insightful comments and feedback to my work. I am grateful to all former and current members of ISMoG group in the Department of Electrical Engineering, a special thanks goes to Hane Aung, Vivek Govinda and Konstantinos Nikolaidis, which made my daily life at our workplace more enjoyable.

Beyond academic matters, I have been blessed with a number of friends that brightened my life in Liverpool. It was a pleasure to be part of the Mexican Students Society in Liverpool, where I found many good friends for life. I want to also thank Daniel Rito, Elihu Loza, and Rolando Medellin for being companionable housemates and helped me to cope with the ups and downs of research life.

Finally my research was fully supported by grant number 196291 of the Mexican Council of Science and Technology CONACYT.

# Acronyms

ANN	Artificial Neural Networks
DNE	Discriminant Neighbourhood Embedding
DONPP	Discriminative ONPP
EA	Evolutionary Algorithms
EKPP	Evolutionary Kernel Projection Pursuit
EPP	Evolutionary Projection Pursuit
FLD	Fisher Linear Discriminant
GA	Genetic Algorithms
GBML	Genetic-Based Machine Learning
GEV	Generalized Extreme Value
GP	Genetic Programming
ICA	Independent Component Analysis
IQR	Inter-Quartile Range
KFD	Kernel Fisher Discriminant
KLPP	Kernel Locality Preserving Projections
kNN	k-Nearest Neighbours
KPCA	Kernel Principal Component Analysis
LDA	Linear Discriminant Analysis
LE	Laplacian Eigenmaps
LLE	Local Linear Embedding
LPP	Locality Preserving Projections
ML	Machine Learning
MMC	Maximum Margin Criterion
MP	Mathematica Programming
NLP	Non-linear Programming
OLPP	Orthogonal LPP
OLPP-R	OLPP with repulsion Laplacian
ONPP	Orthogonal Neighbourhood Preserving Projections
ONPP-R	ONPP with repulsion Laplacian
PCA	Principal Component Analysis
POCS	Projection Onto Convex Sets
PP	Projection Pursuit
PPP	Parallel Projection Pursuit
PSO	Particle Swarm Optimisation
SPP	Sequential Projection Pursuit
SSE	Supervised Spectral Embedding
SONPP	Supervised ONPP
SVM	Support Vector Machine
USE	Unsupervised Spectral Embedding

# Mathematical Notation

## Pattern Classification

$\mathbb{R}^m$	Original feature space
$\mathbf{Y} = \{-1, +1\}$	Binary label space
$n$	Number of instances
$\mathbf{x}_i$	Feature vector in $\mathbb{R}^m$
$(\mathbf{x}_i, y_i)$	Feature vector and its corresponding class label
$\psi : \mathbb{R}^m \rightarrow \mathbf{Y}$	Classification mapping
$\psi(\mathbf{x}), \psi_+(\mathbf{x})$	Separating hyperplane
$\mathbf{p}, b$	Weight vector and bias of $\psi(\mathbf{x})$
$p(\mathbf{x}, y)$	Joint probability of an example $(\mathbf{x}, y)$
$p(y \mathbf{x})$	Class conditional probability, also known as posterior probability
$p(y = +1)$	Probability of the positive class
$p(y = -1)$	Probability of the negative class
$N(\mu, \Sigma)$	Normal distribution with mean $\mu$ and covariance matrix $\Sigma$
$k_n$	Number of samples inside an hypercube
$u_{jk}$	Refers to the $j^{\text{th}}$ input on the $k^{\text{th}}$ layer of an ANN
$f_{ik}$	Activation function of the $i^{\text{th}}$ neurone in the $k^{\text{th}}$ layer of an ANN
$w_{ij}^{(k)}$	Weight for the input $u_{jk}$ going to the activation function $f_{ik}$

## Feature Extraction

$\Upsilon = \{(\mathbf{x}_i, y_i)\}$	Learning set
$\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^T$	Matrix representation of a set of $n$ feature vectors
$\mathbf{Y} = [y_1, \dots, y_n]$	Matrix representation of a set of $n$ class labels corresponding to each row in matrix $\mathbf{X}$
$\mathbf{P} = [\mathbf{p}_1, \dots, \mathbf{p}_b]$	Projection matrix that reduce the dimensionality of a set of $m$ -dimensional feature vectors by means of the linear transform $\mathbf{Z} = \mathbf{XP}$
$\mathbf{x}_l$	Testing point in the original feature space
$\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_n]^T$	Projected version of $\mathbf{X}$ in a $b$ -dimensional space
$\mathbf{S}_T$	Total scatter matrix
$\mathbf{S}_W$	Within-class scatter matrix
$\mathbf{S}_B$	Between-class scatter matrix
$\lambda_i$	$i$ -th eigenvalue corresponding to the $i$ -th eigenvector that is a solution to a given generalized eigenvalue problem

$\mathbf{M}$	Inverse of the projection matrix $\mathbf{P}$
$E\{\cdot\}$	Mathematical expectation
$\mathbf{v}$	Vector orthogonal to $\mathbf{p}_i$
$\mathbf{E}$	Modal matrix
$\phi : \mathbb{R}^m \rightarrow \mathcal{H}$	Non-linear mapping to a non-observable high-dimensional space
$\Phi = [\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_n)]^T$	Matrix representation of the non-linear projected version of the feature vectors in $\mathbf{X}$
$K(\cdot, \cdot)$	Kernel function
$\mathbf{K}$	Kernel matrix of the feature vectors in $\mathbf{X}$
$\gamma$	Coefficient set defining a basis in $\mathcal{H}$
$\mathbf{K}_l$	Row vector of pairwise inner products between $\mathbf{x}_l$ and $\mathbf{X}$ in the non-observable space $\mathcal{H}$
$\bar{\mathbf{K}}$	Kernel matrix of the centred data in $\mathcal{H}$
$\mathbf{I}_n$	Identity matrix of size $n \times n$
$\mathbb{I}_n$	Square matrix of size $n \times n$ with all its elements equal to $1/n$
$\mathbf{1}_n$	Column vector with $n$ elements equal to one
$v_i$	$i^{\text{th}}$ class mean in the non-observable space
$\mu_i, \Sigma_i$	$i^{\text{th}}$ class mean and covariance matrix in the original feature space
$n_i$	Number of samples in the $i^{\text{th}}$ class
$\check{\mathbf{K}}$	Ordered version of $\mathbf{K}$ , with its rows and columns sorted according to class labels
$\text{Ind}(y = j)$	Set of indices for samples belonging to the $i^{\text{th}}$ class
$\mathfrak{S}(\cdot)$	Projection pursuit index
$\mathbf{P}^* = [\mathbf{p}_1^*, \dots, \mathbf{p}_b^*]$	Optimal projection matrix in PP with $b$ projection vectors as its columns
$\mathbf{S}_j^\perp$	Orthogonal complement to the first $j^{\text{th}}$ optimal projection vectors
$B$	Modified Bayes factor
$\mathbf{A}$	Arbitrary weight matrix
$\mathbf{B}$	Arbitrary scaling/constraint matrix
$\mathbf{W}$	Similarity matrix
$\mathbf{G}$	Local covariance matrix
$\mathbf{D}(\mathbf{W})$	Degree matrix of $(\mathbf{W})$
$\mathbf{L}$	Laplacian matrix
$\tilde{\mathbf{S}}_W, \tilde{\mathbf{S}}_B$	Local within-class and between-class scatter matrices
$\tilde{\mathbf{W}}$	Block diagonal matrix with regression coefficients as elements of each block
$\mathbf{W}^r$	Weighting matrix for the repulsion graph
$\mathbf{W}^s$	Weighting matrix for the class graph
$\mathbf{W}'$	Binary adjacency matrix

## Optimisation Techniques in Machine Learning

$\mathbf{x}$	$n$ -dimensional vector of design variables
$f(\cdot)$	Objective function
$g_j(\cdot)$	Inequality constraint function
$l_i(\cdot)$	Equality constraint function
$\mathbf{c}$	Weighting column vector
$\mathbf{b}$	Bias column vector
$\tilde{\mathbf{A}}$	Mixing matrix
$\mathbf{U} = [u_{ij}]$	Weighting matrix for the quadratic term, with elements $u_{ij}$
$\lambda_i, \theta_j$	Lagrange multipliers
$\boldsymbol{\lambda} = [\lambda_i], \boldsymbol{\theta} = [\theta_j]$	Column vectors of Lagrange multipliers
$\mathbf{Y}$	Column vector of slack variables
$z_j$	Artificial variables
$\mathbf{x}^*$	Optimum value for the design variables
$\frac{\partial f}{\partial x_i}$	Partial derivative of $f$ with respect to variable $x_i$
$\mathbf{J}_{\mathbf{x}^*}$	Hessian matrix of $f$ evaluated at $\mathbf{x}^*$
$\nabla f$	Gradient vector of partial derivatives of $f$
$\mathbf{S}_i$	Steepest descent direction of $f$ at point $\mathbf{x}_i$
$\varpi_i^*$	Optimal displacement constant at point $\mathbf{x}_i$
$\mathbf{A}_i$	Approximation to the Hessian matrix of $f$ at point $\mathbf{x}_i$
$\mathbf{B}_i$	Approximation to the inverse of the Hessian of $f$ at point $\mathbf{x}_i$
$\mathbf{g}_i$	Column vector of the same dimensions as $\nabla f$ , indicating the variation of the gradient
$\mathbf{d}_i$	Optimal displacement vector pointing to the steepest descent direction

## Evolutionary Optimisation

$P_c$	Crossover probability
$P_m$	Mutation probability
$N_{gen}$	Total number of generations
$N_{ind}$	Total number of individuals in the population
$\mathcal{P}_t(i, k)$	$k^{th}$ allele of the $i^{th}$ member of the population at generation $t$
$L_k, U_k$	Lower and upper bound for the values allowed for the $k^{th}$ allele
$\Omega$	Adaptive, random-generator function
$F_{best}$	Fitness value of the best individual in the current population
$F_{avg}$	Average fitness value of the current population
$F_k$	Average fitness value at generation $k$
$N_{stall}$	Window size for stagnation test
$f(\cdot)$	Fitness function
$\rho$	Generational gap ratio
$S_{best}(k)$	Set of fittest individuals at generation $k$
$d$	Tree depth
$d_{max}$	Maximum tree depth
$N$	Number of particles for PSO
$\mathbf{x}_i$	Position of the $i^{th}$ particle
$\mathbf{v}_i$	Velocity vector of the $i^{th}$ particle
$\mathbf{p}_i$	Best position of the $i^{th}$ particle

$\mathbf{p}_g$	Global best position of the swarm
$c_1, c_2, w$	Constant coefficients and inertia weight
$z$	Random positive number

## Chapter 4 Automated Induction of Projection Pursuit Indices

$\{\mathfrak{S}^* \theta^*\}$	Optimal pair of projection index and hyperparameters
$F(\cdot, \cdot; \cdot)$	Fitness function for model induction
$L(\cdot, \cdot)$	0-1 loss function
$\Omega_i, \tilde{\Omega}_i$	Training and validation sets for the $i^{th}$ fold in cross-validation
$\varphi_{\Omega_i}$	Classifier function trained with prototypes in $\Omega_i$
$\theta = \{k, \lambda\}$	Classifier hyperparameters corresponding to number of neighbours and Minkowski distance coefficient
$\mathbf{p}_i^{(j)}$	Deflated version of the $i^{th}$ optimal projection vector
$\hat{f}_i$	Projected data probability estimate evaluated at $\mathbf{z}_i$
$g_i$	Reference pdf evaluated at $\mathbf{x}_i$
$D(\hat{f}, g, \rho)$	Rényi's generalized divergence of order- $\rho$
$H(\hat{f}; \rho)$	Rényi order- $\rho$ entropy
$m_s$	$s^{th}$ central sample moment
$\mu_c, \sigma_c$	$c^{th}$ class mean and variance of the projected data $\mathbf{z}$
$\mathbf{Q}_n^j$	$n^{th}$ quartile of class $j$ of $\mathbf{z}$
$S_W, S_B$	Within-class and between-class scatter of $\mathbf{z}$
$\tau$	Scalar indicating the reference pdf $g(\mathbf{x})$ to be used
$\nu$	Degrees of freedom for Student-t distribution
$\xi$	Shape parameter for GEV distribution
$\eta$	Ephemeral random constant
$N_{feat}, \text{Feat}$	Number of reduced features
$T_1$	Average time for feature extraction and classification
$T_2$	Average time for index evolution
$D = \Upsilon \cup \tilde{\Upsilon}$	Dataset partitioned into learning set and testing set
$w_k$	Weighting coefficients for collaborative index
$A_z$	Area under ROC

## Chapter 5 Nonlinear Projection Pursuit via Kernel-induced Spaces

$\mathcal{M}$	Original feature space
$\mathcal{B}$	Reduced feature space
$\nu$	Projection vector in $\mathcal{H}$
$\nu^*$	Optimal projection vector in $\mathcal{H}$
$\gamma^*$	Optimal coefficient set defining a basis in $\mathcal{H}$ for KPP
$\mathbf{K}_i$	Kernel matrix between samples from $\Omega_i$
$\tilde{\mathbf{K}}_{ji}$	Kernel matrix between the $j^{th}$ sample from the validation set $\tilde{\Omega}_i$ and all the samples from the training set $\Omega_i$

$\mathbf{V} = [\boldsymbol{\nu}_1, \dots, \boldsymbol{\nu}_b]$	Projection matrix in $\mathcal{H}$
$\boldsymbol{\Gamma} = [\gamma_1, \dots, \gamma_b]$	Coefficient matrix to express $\mathbf{V}$ in terms of a basis
$\boldsymbol{\nu}_j^*$	$j^{\text{th}}$ optimal projection vector in $\mathcal{H}$
$\mathbf{V}_j^*$	Set of the first $j$ optimal projection vectors obtained by SPP
$\boldsymbol{\Gamma}_j^*$	Optimal coefficient matrix for $\mathbf{V}_j^*$
$\mathbf{K}^{(j+1)}$	Residual kernel matrix to compute the coefficient set $\gamma_{(j+1)}^*$
$\mathbf{C}_\Phi$	Covariance matrix in $\mathcal{H}$
$\mathbf{W}$	Whitening matrix in $\mathcal{H}$
$\mathbf{M}$	Modal matrix of $\mathbf{C}_\Phi$
$\mathbf{L}$	Diagonal matrix with the eigenvalues of $\mathbf{C}_\Phi$ on its diagonal
$\boldsymbol{\Psi}$	Modal matrix of $\mathbf{K}$
$\boldsymbol{\Lambda}$	Diagonal matrix with the eigenvalues of $\mathbf{K}$ on its diagonal
$\hat{\mathbf{K}}$	Whitened kernel matrix
l-SVM	Linear SVM classifier
g-SVM	SVM classifier with Gaussian kernel
MI+g-SVM	SVM classifier with Gaussian kernel, and input feature selected by mutual information
$\theta_1, \theta_2, \theta_3$	Kernel parameters

## Chapter 6 Evolutionary Induction of Heterogeneous Proximities for Supervised Embeddings

$N_F(\mathbf{x}_i, k)$	Set of $k$ -nearest friends for sample $\mathbf{x}_i$
$N_E(\mathbf{x}_i, k)$	Set of $k$ -nearest enemies for sample $\mathbf{x}_i$
$\mathbf{M} = [m_{ij}]$	Reconstruction coefficient matrix
$\beta$	Penalisation coefficient
$f_l(\cdot)$	Similarity measure for the $l^{\text{th}}$ on-diagonal block of the composite weight matrix
$g_{pq}(\cdot)$	Similarity measure between enemies of classes $p$ and $q$
$N_{E\mathbf{x}_i, k_{pq}, q}$	Set of $k_{pq}$ -nearest enemies for sample $\mathbf{x}_i$ from class $q$
$\mathbf{F}$	Proximity matrix between friends
$\mathbf{G}$	Proximity matrix between enemies
$\tilde{\mathbf{F}}$	Laplacian form of $\mathbf{F}$
$\tilde{\mathbf{G}}$	Laplacian form of $\mathbf{G}$
$\lambda$	Regularization parameter
$N_d$	Cardinality of a set of similarity functions
$\mathbf{C}_m$	Centring matrix of size $m \times m$
$\mathbf{X}_l$	Set of points belonging to class $l$
$\mathbf{F}_l$	Proximity matrix between $\mathbf{X}_l$
$\mathcal{F}_l, \mathcal{G}_{pq}$	Matrix representation of $f_l(\cdot)$ and $g_{pq}(\cdot)$

# Chapter 1

## Introduction

### 1.1 Motivation and Objectives

The importance of pattern classification and data analysis has been studied and understood through the last few decades. R. Duda [1] in the introduction of his book offers a good example of how pattern classification can be applied to solve real-world problems, allowing the reader to distinguish the different levels in which he decomposes a full pattern recognition system following a bottom-up approach. Following his discussion, feature extraction is described as a searching process in order to find “... distinguishing features that are invariant to irrelevant transformations ...” [1].

According to this author, the distinction between feature extraction and classification is somehow blurred by the fact that both stages in pattern recognition systems tend to interact closely. Ideally a complex feature extractor would deliver a representation robust enough to simplify the classification task; on the contrary, a powerful classifier would not need the help of a feature extraction stage. In the real world, both stages are needed since there is not such a thing as an infallible classifier or a perfect feature extractor. Therefore the feature extraction task has been redefined as to search for a suitable transformations that allow redundancy reduction over the space spanned by the original features [2].

On the other hand, a simple phenomenon can be measured in several ways, which may lead to a multiple representation of the same phenomenon but in different domains, therefore multidimensional features vectors can be used to process such information. Two close related problems can be experienced when working with multivariate data: (1) the so-called Hughes phenomenon, that states a direct linear relation between the number of dimensions and the number of samples needed to achieve high classification accuracy [2]; and (2) the curse of dimensionality, saying that for nonparametric classifiers based on density estimation, as the number of dimensions in the feature vectors increases, the sample size needs to increase exponentially to successfully obtain a good estimate.

As an example lets consider a k-nearest neighbours classifier in a  $N$ -dimensional



space. To capture a fraction  $r$  of the unit volume around a specific point, we need a subcube with side-length given by  $e_p(r) = r^{1/p}$ . For 10 dimensions, to capture 1% of the data to perform the k-nearest neighbours procedure, we need to cover  $e_{10}(0.01) = 0.63$  of the unit range of each input, turning out to be a non-so-local neighbourhood. Besides, the computational cost of handling high-dimensional data in common matrix operations frequently used in machine learning and pattern classification is extremely high. Therefore the interest in searching for new methods that allow us to transform high-dimensional spaces into low-dimensional ones but keeping the information needed to accomplish successfully our task.

The principal objective of feature extraction is to derive new features from the direct measurements of an experiment in order to boost classification accuracy and classifier efficiency. The problem of finding a suitable transform for the raw data able to deliver invariant features is not easy. The standard way to perform feature extraction is through linear transformations, since they are easy to compute and exhibit useful properties such as statistical invariance, high information packing and redundancy removal. Among linear transformations in pattern classification, projection techniques such as principal component analysis (PCA), linear fisher discriminant (LFD), and independent component analysis (ICA) have been widely used in a variety of applications like brain imaging [3], telecommunications [4], audio separation and financial applications [5].

Despite the advances in pattern recognition and feature extraction, according to the *no free lunch* theorem [6], there is no single algorithm that can be optimal for every classification problem. In practice, two alternatives exist to propose adequate feature extraction and classification algorithms when facing a new classification problem. The first one is the selection of the best performing pair among the existing algorithms. Frequently, such selection process is modelled as a computationally expensive grid search over all possible feature-extraction/classifier pairs, involving parameter fine-tuning for each algorithm [7]. Although better search techniques have been developed to tackle the computational burden of grid search [8–10], the resulting model may still be suboptimal as it makes strong assumptions about the characteristics of the problem and the data distribution. The second option is to specifically design a new feature-extraction/classification pair tailored to the problem at hand. Traditionally, this task involves human experts which analyse the data, characterise the problem and eventually propose a new mathematical model. The objective of this work is to propose a human-competitive alternative in the design of projection methods to extract discriminative features that will assist in the problem of classification.

## 1.2 Scope and Contributions

The principal contribution of this work lies on the introduction of an evolutionary framework to solve the problem of tailoring a feature extraction method for a given classification problem. Three areas of feature extraction are explored: linear projection methods, kernel-induced spaces for nonlinear projection methods, and spectral embedding methods. Among the existing evolutionary algorithms, genetic programming (GP) was selected to jointly perform model search and parameter optimisation of a feature-extraction/classifier pair. The flexible tree-based chromosome encoding in GP lends itself to the specific needs of the aforementioned feature extraction areas, allowing us to encode a feature-extraction/classifier pair into a single individual. Thus, representation is one of the key issues addressed in this work, and it will drive our discussion in the upcoming chapters.

Although heavily based on evolutionary algorithms, this work does not intend to explore new genetic operators, nor compare different fitness landscapes provided by different objective functions. Instead, experimental evidence is provided to support the idea that carefully designed evolutionary algorithms, when applied to a given classification problem, possess enough expressive power to not only solve the problem of model selection, but to create new models when the existing ones perform poorly. The design of our evolutionary framework consists of three main components: a composite chromosome able to capture the main characteristics of the targeted feature extraction area; a fitness function to measure the performance of each potential model; and a function and terminal sets that provide the primitives or building blocks for the construction of potential models. A summary of the contributions of each chapter is provided in the remaining of this section, focusing on the mentioned components.

**Chapter 4.** This chapter introduces the proposed evolutionary inducer, posing the problem of model learning as a complex inference task, where a population of potential solutions are evolved by means of genetic operators. A novel solution encoding, regarded as an hybrid chromosome, was designed to hold a tree-based part representing the projection index, and a scalar part with the parameters needed by the tree-based part, and by the targeted classifier. Using the proposed inducer, the problems related to design of linear projection methods are tackled by developing a generalization framework based on projection pursuit. Sustained by a carefully designed set of function primitives, the proposed inducer creates a new projection pursuit index which delivers an optimal feature extraction stage for the problem at hand. Additionally, it overcomes the problem of manually characterising an interesting projection, when no information regarding the underlying structure of the targeted dataset is available.

**Chapter 5.** A sequential model for nonlinear projection pursuit is presented, which optimises an evolutionary index in a kernel-induced feature space. The evolutionary index is obtained through the evolutionary framework described in Chapter 4. Determination of a nonlinear residual subspace for sequential projection pursuit is reduced to the computation of an updated kernel matrix. Additionally, given the use of whitening as a preprocessing stage in sequential projection pursuit, this chapter provides a kernel extension for whitening in feature space.

**Chapter 6.** This chapter introduces a generalised model for supervised spectral embedding, based on a heterogeneous proximity matrix, where the relations between enemies and friends are described through a set of different similarity metrics. Additionally a matrix representation is proposed for each of the aforesaid similarity metrics, which besides providing a quick way of computing blocks of pairwise similarities, promotes compactness and parsimony of the solutions. Different from the previously designed hybrid chromosomes, the proposed composite weight matrix was encoded as a multi-gene chromosome, where each gene consists of a tree-based part and a scalar part representing a given similarity function and a neighbourhood size.

### 1.3 Published/Submitted Articles

In this section the articles published or submitted as a result of the work presented in this thesis are listed.

#### 1. Journal articles.

- E. Rodriguez-Martinez, J. Y. Goulermas, Tingting Mu, and J. F. Ralph, "Automatic Induction of Projection Pursuit Indices", *IEEE Transactions on Neural Networks*, vol. 21, no. 8, pp. 1281-1295, 2010.
- E. Rodriguez, K. Nikolaidis, T. Mu, J. F. Ralph, and J. Y. Goulermas, "Towards Collaborative Feature Extraction for Face Recognition", *Natural Computing*, DOI 10.1007/s11047-011-9285-6, in press.
- E. Rodriguez-Martinez, J. Y. Goulermas, Tingting Mu, and J. F. Ralph, "Evolutionary Kernel Projection Pursuit for Supervised Feature Extraction", submitted to *IEEE Transactions on Systems, Man, and Cybernetics, Part B*.
- E. Rodriguez-Martinez, Tingting Mu, J. Jiang, and J. Y. Goulermas, "Evolutionary Induction of Heterogeneous Proximities for Supervised Embedding", submitted to *IEEE Transactions on Pattern Analysis and Machine Learning*.

2. Conference articles.

- E. Rodriguez, K. Nikolaidis, T. Mu, J. F. Ralph, and J. Y. Goulermas, "Collaborative Projection Pursuit for Face Recognition", in Proc. IEEE 5th Int. Conf. Bio-Inspired Computing: Theories and Applications, pp. 1346-1350, Sept. 2010.

## Chapter 2

# Machine Learning Methodologies

Artificial intelligence is a broad field of study that aims at developing intelligent agents capable of mimicking human reasoning and able to interact with their environment when performing a specific task. One characteristic such agents must possess is the ability to learn, understood as an adaptation mechanism that modifies a specific behaviour when performing a given task, that allows an increase in the agent's performance on repetition of the same task. From this idea, *machine learning* (ML) sprang as the discipline concerned with the implementation and development of learning algorithms. Nevertheless, modelling the learning process not only requires defining an action plan to modify an agent's behaviour, known as learning strategy, but also needs to define source and representation of the training experience, and a measure of performance.

Being a multidisciplinary field, ML makes use of significant results and postulates of a wide range of subjects such as philosophy, information theory, probability and statistics, control theory, optimisation theory, and computational complexity theory. With such portfolio of subjects, ML features in a variety of applications such as natural language processing [11], handwriting recognition [12], face and fingerprint recognition [13–15], search engines [16], medical diagnosis [17, 18], bioinformatics and cheminformatics [19–21], detecting credit card fraud [22], stock market analysis [23], classification of DNA sequences [24], object recognition in computer vision [25], image compression [26–28], game playing [29], robot locomotion [30], and machine condition monitoring [31, 32].

### 2.1 Types of Algorithms

The representation of the training experience for a given learning problem indirectly defines the type of learning strategy that must be adopted. In ML, algorithms are cataloged according to the representation of the target function into supervised learning, unsupervised learning, reinforcement learning, and transduction.

### 2.1.1 Supervised Learning

It is the most common type of learning algorithms, it is based on the assumption that training examples are given in the form of descriptive feature vectors paired with their desired outputs [33]. The target function is represented by a mapping from the feature space to the output domain. Supervised learning tries to reduce the error between the estimated target for a given training sample, and its desired output. Once the target function is learnt, it can be used to estimate the category of a previously unknown example. When the target is a continuous signal, the task is known as regression, and when it is a label describing the class to which the input vector belongs, it is called classification.

### 2.1.2 Unsupervised Learning

Unsupervised learning [34] is an inference process in which implicit relationships among training examples are uncovered. In unsupervised learning there is no a priori information about the output. The input examples are grouped following a similarity criterion, which can be implicitly defined in the algorithm or explicitly given by a similarity function. Generally, the only a priori information an unsupervised algorithm needs is an estimate of the number of groups defining the data structure. Representative examples of unsupervised learning are clustering and dimensionality reduction.

### 2.1.3 Reinforcement Learning

Reinforcement learning [35] studies the interaction between the agent and its environment, in order to develop strategic plans such that any action performed by the agent to modify its environment contributes towards improving a long-term reward. Generally, a reinforcement learning problem is modelled as a Markov decision process (MDP) composed of four elements: a finite set of states, a finite set of actions, a probability transition matrix  $\mathbf{P}$ , and a reward transition matrix  $\mathbf{R}$ . Its solution involves maximising a weighted sum of rewards given by an optimal policy. A policy is a mapping from a given state to a specific action. A MDP is solved using dynamic programming assuming a priori knowledge of  $\mathbf{P}$  and  $\mathbf{R}$  by means of two variations: policy iteration which starts with a initial policy and iteratively improves it, and value iteration which is based on the convergence of a set of values known as  $Q$ -values. When no a priori information is assumed, temporal difference methods are used to learn an optimal policy. There are two principal algorithms for temporal difference: actor-critic learning which parallels policy iteration, and  $Q$ -learning which parallels value iteration.

### 2.1.4 Transduction

Different from induction, where a general rule to classify future examples is learnt from training samples, transduction propose to directly predict the class label of a given sam-

ple based only on the available training examples [36]. Since it lacks a predictive model, if a previously unknown sample is added to the problem, the whole algorithm would need to be run again to produce a label. On the other hand, transductive algorithms may be able to make better predictions with fewer points. When the objective is to assign a discrete label to unlabelled points, the design of a transducer involves adding partial-supervision to a clustering algorithm. According to the clustering technique, such algorithms can be further divided into partitioning transduction and agglomerative transduction. When the target is a continuous label, a transducer is designed by adding partial-supervision to a manifold learning algorithm.

## 2.2 Pattern Classification

Classification can be placed at the core of the decision process in an intelligent agent. It is the result of applying previously learned rules to sensory inputs. Typically, such rules are modelled as a classification function whose parameters are estimated using supervised learning. The classification task can be divided into two groups: binary classification when the task consists in discriminating among two groups, and multi-class classification when there exists more than two categories. The binary classification framework can be extended to solve multi-class problems. Let the feature vectors be labelled with one of  $c$  class labels, thus the task can be broken into  $c$  one-against-all binary classification problems, each of which builds a classifier separating one class from the rest. Formally, binary classification can be defined as follows: Given a set of training examples  $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$  with  $n$   $m$ -dimensional feature vectors  $\mathbf{x}_i \in \mathbb{R}^m$  paired to  $n$  corresponding class labels  $y_i \in \mathbf{Y} = \{-1, +1\}$ , the task consists in approximating a unknown classification function  $\psi : \mathbb{R}^m \rightarrow \mathbf{Y}$  that best predicts the label for an input sample.

### 2.2.1 Linear Classifiers

Linear classifiers take their name from the linear function used to model the decision boundary, which is modelled as a hyperplane in the  $m$ -dimensional feature space as

$$\psi(\mathbf{x}) = \mathbf{p}^T \mathbf{x} + b, \tag{2.1}$$

where  $\mathbf{p} \in \mathbb{R}^m$  is a weight vector and  $b \in \mathbb{R}$  is a threshold weight or bias. These parameters are estimated from training examples using techniques such as maximum likelihood, maximum a posteriori probability, Bayesian inference, or expectation maximisation. Usually, when a Bayesian approach is used to model the decision process, Eq. (2.1) is given by the a posteriori probability  $p(y|\mathbf{x})$ . Based on how such probability is estimated, linear classifiers are divided into generative and discriminative.

### 2.2.1.1 Generative classifiers

Generative classifiers [37,38] use Bayesian decision theory to guarantee optimality in the hyperplane parameters. They model a joint probability  $p(\mathbf{x}, y)$  by means of Bayes rule as  $p(\mathbf{x}|y)p(y)$  and learn the model parameters through maximisation of the a posteriori probability. Their name comes from the fact new samples can be generated using the estimated a priori and class-conditional probabilities. The decision rule selects  $y = +1$  if  $p(y = +1|\mathbf{x}) > p(y = -1|\mathbf{x})$ , otherwise  $y = -1$ . Such rule has been proven to minimise both average risk and classification error probability. From a mathematical point of view, sometimes it is better to work with a monotonically increasing function of the probabilities, such that hyperplane separating class +1 from class -1 can be modelled as

$$\ln \left( \frac{p(\mathbf{x}|y = +1)}{p(\mathbf{x}|y = -1)} \right) + \ln \left( \frac{p(y = +1)}{p(y = -1)} \right) = 0. \quad (2.2)$$

Usually, the class-conditional probability is assumed to be Gaussian and independent. Hence, for class +1 we have  $p(\mathbf{x}|y = +1) \sim N(\mu_+, \Sigma_+)$ , where  $N(\mu_+, \Sigma_+)$  is a Gaussian probability distribution with mean  $\mu_+$  and covariance matrix  $\Sigma_+$ , and its respective discriminant function is

$$\begin{aligned} \psi_+(\mathbf{x}) &= \ln p(\mathbf{x}|y = +1) + \ln p(y = +1) \\ &= -\frac{1}{2}(\mathbf{x} - \mu_+)^T \Sigma_+^{-1} (\mathbf{x} - \mu_+) - \frac{m}{2} \ln 2\pi - \frac{1}{2} \ln |\Sigma_+| + \ln p(y = +1). \end{aligned} \quad (2.3)$$

In general, this is a nonlinear quadratic form and the resulting Bayesian classifier is known as quadratic classifier.

When the covariance matrix is the same in all classes,  $\Sigma_+ = \Sigma_- = \Sigma$ , the quadratic contribution of the term  $\mathbf{x}^T \Sigma \mathbf{x}$ , as well as the constant term  $-\frac{m}{2} \ln 2\pi - \frac{1}{2} \ln |\Sigma|$  will be the same in all discriminant functions and eventually cancelled out in the hyperplane function. Thus the discriminant function can be expressed as

$$\psi_+(\mathbf{x}) = \mathbf{p}_+^T \mathbf{x} + b \quad (2.4)$$

where  $\mathbf{p}_+ = \Sigma^{-1} \mu_+$  and  $b = -\frac{1}{2} \mu_+^T \Sigma^{-1} \mu_+ + \ln p(y = +1)$ . From the structure of the covariance matrix, two cases can be separated:

- Assuming statistically independent features with the same variance  $\sigma$ , the covariance matrix takes the form  $\Sigma = \sigma^2 \mathbf{I}_m$ , with inverse  $\Sigma^{-1} = 1/\sigma^2 \mathbf{I}_m$ , where  $\mathbf{I}_m$  is a  $m \times m$  identity matrix. The weight vector and bias in Eq. (2.4) simplify to  $\mathbf{p}_+ = \frac{1}{\sigma^2} \mu_+$  and  $b = -\frac{1}{2\sigma^2} \mu_+^T \mu_+ + \ln p(y = +1)$ . As for the hyperplane, it can be expressed as well as a linear function of the inputs as follows

$$\mathbf{p}^T (\mathbf{x} - \mathbf{x}_0) = 0, \quad (2.5)$$

where  $\mathbf{p} = \mu_+ - \mu_-$  and  $\mathbf{x}_0 = \frac{1}{2}(\mu_+ + \mu_-) - \sigma^2 \ln \left( \frac{p(y=+1)}{p(y=-1)} \right) \frac{\mu_+ - \mu_-}{|\mu_+ - \mu_-|^2}$



- A second case arises when the classes present the same arbitrary covariance matrix. In this case the discriminant function is the same as in Eq. (2.4), and the hyperplane is described by Eq. (2.5) with parameters  $\mathbf{p} = \Sigma^{-1}(\mu_+ - \mu_-)$  and  $\mathbf{x}_0 = \frac{1}{2}(\mu_+ + \mu_-) - \ln\left(\frac{p(y=+1)}{p(y=-1)}\right) \frac{\mu_+ - \mu_-}{(\mu_+ - \mu_-)^T \Sigma^{-1}(\mu_+ - \mu_-)}$ .

If the parameters of the normal distribution are unknown, they can be estimated from the training data using maximum likelihood (ML), maximum a posteriori probability (MAP), or Bayesian inference estimation. Examples of algorithms in this category are Naive Bayes classifier and Linear Discriminant Analysis.

### 2.2.1.2 Discriminative classifiers

Contrary to generative classifiers, that need the class conditional probability and the a priori probabilities to model the a posteriori probability, discriminative classifiers solve a more specific problem by directly estimating the a posteriori probability  $p(y|\mathbf{x})$  [37, 38]. An advantage of discriminative classifiers is that the number of adaptive parameters to be determined is linearly dependent on the input space dimensionality, thus they perform better than generative classifiers for high dimensional datasets.

An example of discriminative classifiers is logistic regression, where the posterior probability is modelled as follows

$$p(y = -1|\mathbf{x}) = \frac{1}{1 + \exp(\mathbf{p}^T \mathbf{x} + b)} \quad (2.6)$$

and

$$p(y = +1|\mathbf{x}) = \frac{\exp(\mathbf{p}^T \mathbf{x} + b)}{1 + \exp(\mathbf{p}^T \mathbf{x} + b)}. \quad (2.7)$$

Thus, the decision rule can be expressed as the inequality  $\frac{p(y=+1|\mathbf{x})}{p(y=-1|\mathbf{x})} > 1$ . Substituting the definitions of the posterior probabilities and taking the natural logarithm of both sides of the inequality leads to Eq. (2.1), where the label  $y = +1$  is assigned if  $\psi(\mathbf{x}) > 0$ , and  $y = -1$  otherwise.

A second subcategory of discriminant classifiers are those which use a geometric approach to directly estimates the unknown classification function  $\psi(\mathbf{x})$  such as the perceptron algorithm, or support vector machines (SVM). They do not provide posterior probabilities, instead they pose the problem of learning the parameters in Eq. (2.1) as a convex optimisation problem which is solved by means of quadratic programming or Lagrange multipliers.

**Support vector machines** SVMs [39–41] construct a linear separating hyperplane in a given feature space that maximises the distance between itself and any training sample. When a perfect decision surface is assumed, the derived classifier is known as *hard-margin SVM classifier*, and the separating hyperplane, defined as  $\psi(\mathbf{x}) = \langle \mathbf{p}, \mathbf{x} \rangle + b$ ,

can be built by minimising the following optimisation risk

$$\begin{aligned} \min_{\mathbf{p}, b} \quad & \frac{1}{2} |\mathbf{p}|^2, \\ \text{s.t.} \quad & y_i (\langle \mathbf{p}, \mathbf{x}_i \rangle + b) \geq 1, \quad i = 1, 2, \dots, n, \end{aligned} \quad (2.8)$$

where  $|\cdot|$  and  $\langle \cdot, \cdot \rangle$ , correspondingly denote vector norm and dot product in the given feature space. A solution to the above constrained optimisation problem is found with the help of Lagrange multipliers  $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \dots, \lambda_n]^T$ , which transform the referred problem into its dual

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_i y_j \lambda_i \lambda_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \\ \text{s.t.} \quad & \sum_{i=1}^n y_i \lambda_i = 0, \\ & \lambda_i > 0, \quad i = 1, 2, \dots, n. \end{aligned} \quad (2.9)$$

Thus, the solution vector is given by the linear combination of the training set  $\mathbf{p} = \sum_{i=1}^n y_i \lambda_i \mathbf{x}_i$ . Those samples with non-zero  $\lambda_i$  are called *support vectors* (SVs).

For the case when an overlap between classes exists, there is no hard-margin defining an optimal decision surface. Introducing a loss term into the risk optimisation problem with the help of slack variables  $\boldsymbol{\xi} = [\xi_1, \xi_2, \dots, \xi_n]^T$ , allows samples violating the constraint  $y_i (\langle \mathbf{p}, \mathbf{x}_i \rangle + b) \geq 1$  to be considered part of the SVs. In this case, the decision surface is built by solving the optimisation problem

$$\begin{aligned} \min_{\mathbf{p}, b} \quad & \frac{1}{2} (|\mathbf{p}|^2 + C \sum_{i=1}^n \xi_i), \\ \text{s.t.} \quad & y_i (\langle \mathbf{p}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, 2, \dots, n, \end{aligned} \quad (2.10)$$

where  $C$  is a positive regularisation parameter set by the user. Again, using the Lagrange multipliers  $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \dots, \lambda_n]^T$ , a dual version of Eq. (2.10) can be written as

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_i y_j \lambda_i \lambda_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \\ \text{s.t.} \quad & \sum_{i=1}^n y_i \lambda_i = 0, \\ & 0 \leq \lambda_i \leq C, \quad i = 1, 2, \dots, n. \end{aligned} \quad (2.11)$$

This type of SVM is also known as *soft-margin C-SVM* or  $L_1$ -SVM. The  $C$  parameter in Eq. (2.10) controls the trade-off between two conflicting objectives: maximisation of the margin, and minimisation of the training error. An alternative soft-margin SVM, known as  $\nu$ -SVM, attempts to directly control the number of margin errors and SVs involved in the decision surface, which is determined by minimising the optimisation

risk

$$\begin{aligned}
\min_{\mathbf{p}, b} \quad & \frac{1}{2} |\mathbf{p}|^2 - \nu \rho + \frac{1}{n} \sum_{i=1}^n \xi_i, \\
\text{s.t.} \quad & y_i (\langle \mathbf{p}, \mathbf{x}_i \rangle + b) \geq \rho - \xi_i, \\
& \xi_i \geq 0, \quad i = 1, 2, \dots, n, \\
& \rho \geq 0,
\end{aligned} \tag{2.12}$$

where  $\rho$  is a constant defining the margin size, and  $\nu \in [0, 1]$  is a user-defined regularisation parameter, which sets an upper bound on the fraction of margin errors, and a lower bound on the fraction of SVs.

## 2.2.2 Nonlinear Classifiers

Different from linear classifiers, the decision boundary built by nonlinear classifiers is an arbitrary hypersurface different from a  $(m - 1)$ -flat in the  $\mathbb{R}^m$  feature space. Similarly to linear classifiers, nonlinear classifiers can also be divided into discriminant and generative classifiers. Examples of nonlinear discriminant classifiers are multilayer perceptrons, radial basis function (RBF) networks and self-organizing maps (SOMs). As for nonlinear generative classifiers, quadratic Bayesian and  $k$ -nearest neighbours (kNNs) are considered representative examples.

### 2.2.2.1 Generative classifiers

The extension of a naive Bayes classifier for the case of an arbitrary covariance matrix for each class is a natural example for a nonlinear generative classifier [42]. As with the linear case, the class conditional probability is assumed to be Gaussian, but this time the quadratic term in the discriminant function can not be dropped, leading into the quadratic classifier

$$\psi_i(\mathbf{x}) = \mathbf{x}^T \mathbf{C}_i \mathbf{x} + \mathbf{w}_i^T \mathbf{x} + \omega_{i0}, \quad i = +, -, \tag{2.13}$$

where  $i = +, -$  denotes class label,  $\mathbf{C}_i = -\frac{1}{2} \Sigma_i^{-1}$ ,  $\mathbf{w}_i = \Sigma_i^{-1} \mu_i$ , and  $\omega_{i0} = -\frac{1}{2} \mu_i^T \Sigma_i^{-1} \mu_i - \frac{1}{2} \ln |\Sigma_i| + \ln p(y = i)$ .

Other example of nonlinear generative classifier is given by the  **$k$ -nearest neighbours classifier** that is based on nonparametric pdf estimation methods [7]. Such methods are used to estimate the class-conditional probability  $p(\mathbf{x}|y)$  when no assumptions about its structure are taken. They are based on an extension to multiple dimensions of the histogram estimation technique, and follow a common model given by

$$p(\mathbf{x}) = \frac{1}{V} \frac{k_n}{n}, \tag{2.14}$$

which represents the constant pdf value inside a hypercube of volume  $V$  holding  $k_n$  out of  $n$  total samples. Specifically, the  $k$ -nearest neighbours method fixes the number of samples to be contained within the hypercube. It works by placing an hypercube

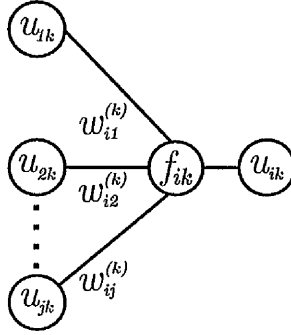


Figure 2.1: Artificial neural network represented as weighted graph.

centred at each training point, which volume is expanded until it covers  $k_n$  points, that will be those closest to the centre. It follows the model

$$p(\mathbf{x}) = \frac{1}{n} \frac{k_n}{V(\mathbf{x})}, \quad (2.15)$$

where the volume dependence on the training point  $\mathbf{x}$  is made explicit by  $V(\mathbf{x})$ . The optimal number of neighbours can be learnt from the training set. The pdf estimate computed by the  $k$ -nearest neighbours can be used within the Bayesian framework to produce the  $k$ -nearest neighbours classifier, which labels a test point with the most frequent class label among its kNNs.

### 2.2.2.2 Discriminative classifiers

The mentioned examples are all based on a mathematical model known as artificial neural networks (ANN) [43]. Inspired by neurobiological analogies, ANN algorithms emulate the way human brains compute, offering highly nonlinear decision surfaces, direct input-output mapping and dynamic adaptive mechanisms. A typical ANN can be regarded as a weighted graph as shown in Fig. 2.1, where each node represents a mathematical model known as artificial neuron, which is defined as

$$u_{i(k+1)} = f_{ik} \left( \sum_j w_{ij}^{(k)} u_{jk} \right), \quad (2.16)$$

where  $u_{jk}$  is the  $j^{th}$  input to the  $i^{th}$  neurone on the  $k^{th}$  layer, weighted by coefficient  $w_{ij}^{(k)}$ . The function  $f_{ik}(\cdot)$  in the above expression is known as activation function, and its purpose is to fire or inhibit the action of the corresponding neurone. Common choices for  $f_{ik}(\cdot)$  are heaviside steps, sigmoids, linear functions, or Gaussians. There are several variations to the structure shown in Fig. 2.1, but the basic directed acyclic graphic (also known as feedforward network) consists of an input layer, one or more (for multilayer perceptrons) hidden layers, and an output layer. When the activation functions in all nodes of the hidden layer depend on the distance to a given point, the

resulting network is known as RBF network. SOMs are an entirely different approach where the weights have a spatial component attached to the weighting coefficients. Such component is used to locate them onto a two-dimensional lattice which produce a low-dimensional representation of the data.

The learning process in an ANN involves adjusting its weights  $w_{ij}^{(k)}$  to minimise a given cost function that depends on the available examples. The selection of a suitable cost function depends on the desired task, and on the properties of the learning algorithm, being the mean-squared error a popular choice for supervising learning. Several learning algorithms have been used to train a neural network, ranging from the classical gradient descent algorithm [43] to more recent and sophisticated evolutionary algorithms [44–46].

## 2.3 Feature Extraction

Selection and design of an ad-hoc classification algorithm plays an important role in classifying an object. Nevertheless, few can be done if the feature vectors describing such object are noisy or present redundant information. Traditionally, in pattern classification systems, a feature extraction stage is used before classification in order to preprocess raw input data, and build representative features. Among feature extraction methods, projection methods are quite popular due to their low computational cost, statistical invariance, and high information packing ability. In the following, a brief description of different projection methods is offered.

### 2.3.1 Projection Methods

Let  $\Upsilon = \{(\mathbf{x}_i, y_i)\}$ ,  $i = 1, \dots, n$ , be a given learning set containing  $n$   $m$ -dimensional whitened samples  $\mathbf{x}_i$ , arranged in a matrix  $\mathbf{X} \in \mathbb{R}^{n \times m}$ , each one belonging to one of  $c$  classes, denoted by  $y_i$  being arranged in a corresponding class-vector  $\mathbf{Y} \in \mathbb{R}^n$ . A linear projection technique generates a projection matrix  $\mathbf{P} \in \mathbb{R}^{m \times b}$  such that

$$\mathbf{z}_i = \mathbf{P}^T \mathbf{x}_i, \quad i = 1, 2, \dots, n \quad (2.17)$$

is a  $b$ -dimensional representation of the  $i^{th}$  sample, for  $b < m$ . When the columns of  $\mathbf{P}$  are orthonormal, they represent the basis of a feature subspace, which can be designed using a given criterion such that the information embedded in each component decreases.

#### 2.3.1.1 Principal component analysis

Principal component analysis (PCA) [47] generates an orthonormal basis spanning a subspace that maximises the total scatter of the projected samples across all categories.

Such basis is obtained by solving the optimisation problem

$$\mathbf{P}_{pca} = \underset{\mathbf{P} \in \mathbb{R}^{m \times b}}{\operatorname{argmax}} \left| \mathbf{P}^T \mathbf{S}_T \mathbf{P} \right| \quad (2.18)$$

where  $\mathbf{S}_T = \sum_{i,j=1}^n \mathbf{x}_i \mathbf{x}_j^T$  is the total scatter matrix. The columns of the optimal basis  $\{\mathbf{p}_i | i = 1, 2, \dots, b\}$  can be estimated by the eigenvectors of  $\mathbf{S}_T$  corresponding to its  $b$  largest eigenvalues.

For high-dimensional data (i.e.  $m \gg n$ ), computing the eigenvectors of the total scatter is computationally expensive. In this case, an alternative method exist based on the singular value decomposition (SVD) of the data matrix  $\mathbf{X}$ . Let

$$\mathbf{X}^T = \mathbf{P} \mathbf{\Sigma} \mathbf{V}^T \quad (2.19)$$

be the SVD of the transpose version of  $\mathbf{X}$ , where  $\mathbf{P}$  is the matrix with columns the eigenvectors of  $\mathbf{X}^T \mathbf{X}$ ,  $\mathbf{V}$  is the matrix with columns the eigenvectors of  $\mathbf{X} \mathbf{X}^T$ . Thus, the PCA projections of  $\mathbf{X}$  can be obtained as

$$\mathbf{Z} = \mathbf{X} \mathbf{P} = \mathbf{V} \mathbf{\Sigma}^T. \quad (2.20)$$

Due to the projection matrix being computed by maximising the total scatter, the extracted features include information regarding the within-class scatter  $\mathbf{S}_W$ . Although such information may provide high compression rate, the resulting subspace does not always provide maximum class-separability as stated in [48].

### 2.3.1.2 Linear Fisher discriminant

In order to diminish the effects of the within-class scatter, and taking advantage of the class-information available to produce higher discriminative features, LFD [1] uses a direct measure of class separability to compute the projection matrix by solving the optimisation problem

$$\mathbf{P}_{lfd} = \underset{\mathbf{P} \in \mathbb{R}^{m \times b}}{\operatorname{argmax}} \frac{\left| \mathbf{P}^T \mathbf{S}_B \mathbf{P} \right|}{\left| \mathbf{P}^T \mathbf{S}_W \mathbf{P} \right|} \quad (2.21)$$

where

$$\mathbf{S}_B = \sum_{i=1}^c n_i (\mu_i - \mu)(\mu_i - \mu)^T \text{ and } \mathbf{S}_W = \sum_{i=1}^c \sum_{\mathbf{x}: y \in i} (\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)^T \quad (2.22)$$

are the between- and within-class scatter matrices,  $\mu_i$  is the average of  $n_i$  samples in the  $i^{\text{th}}$  class, and  $\mu$  is the global mean. This time the columns of the optimal basis  $\{\mathbf{p}_i | i = 1, 2, \dots, b\}$  are estimated by solving the eigenvalue problem  $\lambda_i \mathbf{S}_W \mathbf{p}_i = \mathbf{S}_B \mathbf{p}_i$ ,  $i = 1, 2, \dots, b$  where the set of nonzero generalised eigenvalues  $\{\lambda_i | i = 1, 2, \dots, c-1\}$  impose an upper bound to the number of extracted features.

Given the high-dimensionality of the data, where  $m \gg n$ , the small sample size problem is always present. A popular way to cope with such problem is to apply PCA

in a first stage to reduce the dimensionality of the original space to  $n - c$ , and then applying LFD as defined in Eq. (2.21) to further reduce the dimensions to  $(c - 1)$ . In this way, the resulting search space after PCA dimensionality reduction possesses a nonsingular within-class scatter matrix. Thus, the optimal basis  $\mathbf{P}_{opt}$  is computed as follows

$$\mathbf{P}_{opt} = \mathbf{P}_{pca}\mathbf{P}_{lfd} \quad (2.23)$$

where

$$\mathbf{P}_{lfd} = \underset{\mathbf{P} \in \mathbb{R}^{(n-c) \times b}}{\operatorname{argmax}} \frac{|\mathbf{P}^T \mathbf{P}_{pca}^T \mathbf{S}_B \mathbf{P}_{pca} \mathbf{P}|}{|\mathbf{P}^T \mathbf{P}_{pca}^T \mathbf{S}_W \mathbf{P}_{pca} \mathbf{P}|}$$

and  $\mathbf{P}_{pca}$  is defined as in Eq. (2.18). Note that in this case, the space search for  $\mathbf{P}_{pca}$  consists of all the orthonormal matrices of size  $m \times (n - c)$ , whereas for  $\mathbf{P}_{lfd}$ , it consists of all the orthonormal matrices of size  $(n - c) \times b$ .

### 2.3.1.3 Independent component analysis

Contrary to PCA and LFD, ICA does not count with a close form solution but rather performs an iterative optimisation over a constrained space to maximise a given cost function [49]. The resulting projection matrix maps the original features to a subspace where every component is independent of each other. The core of ICA is the assumption that each dimension of the observed samples  $\mathbf{x}_i$  is a linear combination of  $b$  low-dimensional sources  $\mathbf{z}_i$  given by  $\mathbf{x}_i = \mathbf{M}^T \mathbf{z}_i$ , thus by estimating the components of the mixing matrix  $\mathbf{M} = \mathbf{P}^{-1}$  we could estimate every independent source  $\mathbf{z}_i$ .

Lets consider the linear combination  $\hat{\mathbf{z}}_i = \mathbf{X}\mathbf{p}_i = \mathbf{Z}\mathbf{M}\mathbf{p}_i = \mathbf{Z}\mathbf{q}_i$ , which gives the  $i^{th}$  component of the projected samples onto a low-dimensional subspace. In order for the estimate  $\hat{\mathbf{z}}_i$  to be exactly the correspondent  $i^{th}$  independent component of  $\mathbf{Z}$ ,  $\mathbf{q}_i$  should be a column vector with its  $i^{th}$  element equals to 1 and zeros elsewhere. Given the fact that any combination of the independent components in  $\mathbf{Z}$  is more Gaussian than any of its columns  $\mathbf{z}_i$ , we can find  $\mathbf{p}_i$  by letting it vary until the distribution  $\mathbf{X}\mathbf{p}_i$  reach maximum nongaussianity; at that point  $\mathbf{q}_i$  has only one nonzero component and  $\hat{\mathbf{z}}_i$  is equal to the corresponding  $i^{th}$  column in  $\mathbf{Z}$ . Therefore by looking at the degree of nongaussianity of the projected features distribution we can estimate the components of the transformation matrix as an optimisation process where the search is performed over the space spanned by all the  $\mathbf{p}_i \in \mathbb{R}^m$ .

Several ways to measure the nongaussianity has been proposed such as kurtosis and negentropy. Using gradient based optimisation methods, a search algorithm was proposed in [50] where successive updates based on a contrast function are applied to a random starting point until convergence is met. Such algorithm is known as FastICA

and the updating rules derived for kurtosis and negentropy are as follows

$$\mathbf{p}_i \leftarrow E\{\mathbf{X}^T (\mathbf{X} \mathbf{p}_i)^3\} - 3 \mathbf{p}_i \quad (2.24)$$

$$\mathbf{p}_i \leftarrow E\{\mathbf{X}^T g(\mathbf{X} \mathbf{p}_i) - E\{g'(\mathbf{X} \mathbf{p}_i)\} \mathbf{p}_i\} \quad (2.25)$$

When the distribution of the independent components is Gaussian, no matter how the observed features are projected, the evaluation of the contrast function for any given angle will be the same and FastICA will fail to converge. For a deeper exposition of ICA refer to [49].

FastICA is known as a one-unit algorithm since it estimates only one independent component, i.e. projection onto a one-dimensional space. The way to extend this algorithm to multiple dimensions is by exploiting the fact that the columns in the transformation matrix, corresponding to different transformed features, are orthogonal in the whitened space. Therefore, running a one-unit algorithm several times and applying orthogonalisation over the resulting transformation matrix after every iteration, we can prevent different vectors from converging to the same optimum.

Two major orthogonalisation methods have been developed so far based on Gram-Schmidt orthogonalisation. The first one is called deflationary orthogonalisation, where the components of the transformation matrix are found sequentially as follows:

1. Choose  $b$ , the number of independent components to estimate. Set  $i \leftarrow 1$ .
2. Initialize  $\mathbf{p}_i$
3. Do an iteration of a one-unit algorithm on  $\mathbf{p}_i$
4. Do the following orthogonalisation:

$$\mathbf{v} = \mathbf{p}_i - \sum_{j=1}^{i-1} (\mathbf{p}_i^T \mathbf{p}_j) \mathbf{p}_j \quad (2.26)$$

5. Normalize  $\mathbf{p}_i$  and  $\mathbf{v}$ .
6. If  $\mathbf{p}_i^T \mathbf{v} < 1$ , go back to step 3.
7. Set  $i \leftarrow i + 1$ . If  $i$  is not greater than the desired number of independent components, go back to step 2.

The second one is symmetric orthogonalisation, contrasting with the former one by computing in parallel all the components of the transformation matrix. This parallel processing consists of running  $b$  independent one-unit algorithms, follow by an orthogonalisation involving matrix square roots. Symmetric orthogonalisation consists of the following:

1. Choose  $b$  as the number of independent component to estimate.



2. Initialize randomly the  $\mathbf{p}_i, i = 1, \dots, b$ .
3. Do an iteration of a one-unit algorithm on every  $\mathbf{p}_i$  in parallel.
4. Do a symmetric orthogonalisation as

$$\begin{aligned}\hat{\mathbf{P}} &= (\mathbf{P}\mathbf{P}^T)^{-1/2} \mathbf{P} \\ (\mathbf{P}\mathbf{P}^T)^{-1/2} &= \mathbf{E} \text{diag}(\lambda_1^{-1/2}, \dots, \lambda_b^{-1/2}) \cdot \mathbf{E}^T\end{aligned}\tag{2.27}$$

where  $\mathbf{E}$  is the matrix formed with the corresponding  $b$  eigenvectors from the square matrix  $\mathbf{P}\mathbf{P}^T$ .

5. If  $\hat{\mathbf{P}}^T \mathbf{P} \neq \mathbf{I}_m$ , where  $\mathbf{I}_m$  is a  $m \times m$  identity matrix, go back to step 3.

### 2.3.2 Projection Methods in Kernel-Induced Feature Space

The popularity of linear classifiers over nonlinear classifiers is due to their fast response, robustness in high-dimensional spaces, and minimal storage requirements. Nevertheless, in many real-world problems, a linear decision surface is not enough to distinguish among classes due to the underlying nonlinear data structure. In such cases, a nonlinear mapping can be used to unfold any nonlinearity as can be seen in Fig. 2.2, and then any linear classifier can be used in the new feature space. Additionally, nonlinear projection methods can be used as dimensionality reduction techniques when the number of projection vectors are less than the dimensionality of the input space as described in Eq. (2.17).

#### 2.3.2.1 Kernel principal component analysis

Classical nonlinear feature extraction has been inspired by kernel principal component analysis (KPCA) [51] and the use of the so-called *kernel trick*. KPCA assumes a nonlinear map  $\phi : \mathbb{R}^m \rightarrow \mathcal{H}$  to a non-observable high-dimensional space referred as *feature space*. In such space, bearing in mind centred data, i.e.  $\sum_{j=1}^n \phi(\mathbf{x}_j) = 0$ ,  $\mathbf{x}_j \in \mathbb{R}^m$ , the principal components are computed by solving the eigenvalue problem

$$\lambda \mathbf{p} = \mathbf{C} \mathbf{p},\tag{2.28}$$

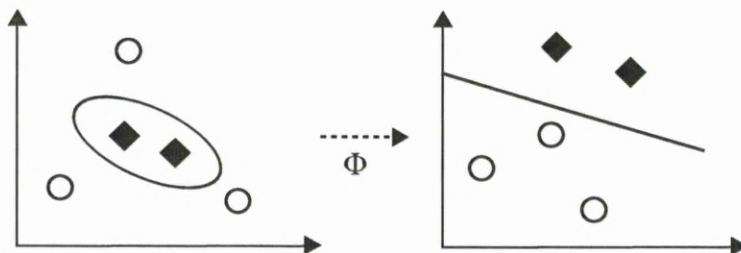


Figure 2.2: The idea of nonlinear mapping is to translate the original feature space into a non-observable feature space where the samples can be linearly separated.

where  $\mathbf{C}$  is the covariance matrix in feature space. As  $\mathbf{C}\mathbf{p} = \frac{1}{n} \sum_{j=1}^n (\phi(\mathbf{x}_j)^T \mathbf{p}) \phi(\mathbf{x}_j)$ , all solutions  $\mathbf{p}$  must lie in the span of  $\{\phi(\mathbf{x}_j)\}_{j=1}^n$ , therefore exist a set of coefficients  $\gamma \in \mathbb{R}^n$  such that  $\mathbf{p} = \Phi^T \gamma$ , where  $\Phi = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \dots, \phi(\mathbf{x}_n)]^T$  is the matrix representation of the mapped data in  $\mathcal{H}$ . Using this expansion, the eigenproblem in Eq. (2.28) can be expressed as

$$n\lambda\gamma = \mathbf{K}\gamma, \quad (2.29)$$

where  $\mathbf{K}$  is a kernel matrix which defines the dot-product among the samples in feature space as  $k_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) = \phi^T(\mathbf{x}_i)\phi(\mathbf{x}_j)$ . Thus, the coordinates of the embedded feature vectors are not needed, but only their pairwise inner product. Once the first nonzero eigenvalues  $\gamma_i, i = 1, \dots, b$  of the kernel matrix have been computed, the  $i^{\text{th}}$  nonlinear principal components of a testing point  $\mathbf{x}_l \in \mathbb{R}^m$  can be obtained as

$$\mathbf{p}_i^T \phi(\mathbf{x}_l) = \gamma_i^T \Phi \phi(\mathbf{x}_l) = \mathbf{K}_l \gamma_i, \quad (2.30)$$

where  $\mathbf{K}_l$  is the  $l^{\text{th}}$  row vector of pairwise inner products between the testing point and the training samples. Then the  $m$ -dimensional testing point is mapped into a  $b$ -dimensional space. When the assumption of centred data in feature space is not valid, the kernel matrix  $\mathbf{K}$  can be replaced by  $\tilde{\mathbf{K}} = \mathbf{K} - \mathbb{1}_n \mathbf{K} - \mathbf{K} \mathbb{1}_n + \mathbb{1}_n \mathbf{K} \mathbb{1}_n$ , where  $\mathbb{1}_n = \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^T$  is a  $n \times n$  matrix with all its elements equal to  $1/n$ , and  $\mathbf{1}_n$  is a column vector with  $n$  elements equal to one.

### 2.3.2.2 Kernel Fisher discriminant

Following KPCA approach of mapping the input data to a non-observable space  $\mathcal{H}$ , kernel Fisher discriminant (KFD) maximises the Rayleigh's ratio between the within-class and the between-class scatter [52], which can be defined in  $\mathcal{H}$  as

$$\mathbf{S}_W = \sum_{i=1}^c \sum_{\forall \phi(\mathbf{x}):y=i} (\phi(\mathbf{x}) - \nu_i)(\phi(\mathbf{x}) - \nu_i)^T \text{ and } \mathbf{S}_B = \sum_{i=1}^c n_i \nu_i \nu_i^T, \quad (2.31)$$

where  $\nu_i = \frac{1}{n_i} \sum_{\forall \phi(\mathbf{x}):y=i} \phi(\mathbf{x})$  is the  $i^{\text{th}}$  class mean. Given the only information regarding  $\mathcal{H}$  is the pairwise inner product among samples in the feature space, the Rayleigh's ratio can be redefined as follows [53]

$$\frac{\mathbf{p}^T \mathbf{S}_B \mathbf{p}}{\mathbf{p}^T \mathbf{S}_W \mathbf{p}} = \frac{\gamma^T \check{\mathbf{K}} \mathbb{I} \check{\mathbf{K}} \gamma}{\gamma^T \check{\mathbf{K}} \check{\mathbf{K}} \gamma} \quad (2.32)$$

where  $\check{\mathbf{K}} = (\check{\mathbf{K}}_{rt})_{r,t=1,\dots,c}$  is an ordered version of  $\mathbf{K}$  composed of blocks  $\check{\mathbf{K}}_{rt} = (k_{ij}), i = 1, \dots, n_r, j = 1, \dots, n_t$  holding the pairwise inner products between samples in class  $r$  and samples in class  $t$ ,  $\mathbb{I} = \text{diag}(\mathbb{1}_{n_1}, \mathbb{1}_{n_2}, \dots, \mathbb{1}_{n_c})$  is a block diagonal matrix, and  $\gamma \in \mathbb{R}^n$  is a set of coefficients that describes  $\mathbf{p}$  within the span of  $\Phi$ . From Eq. (2.32) it is clear that the solution to the optimisation problem posed by Eq. (2.21) in the non-observable feature space is equivalent to the solution of the eigenvalue problem

$$\lambda \check{\mathbf{K}} \check{\mathbf{K}} \gamma = \check{\mathbf{K}} \mathbb{I} \check{\mathbf{K}} \gamma, \quad (2.33)$$

and the  $i^{th}$  nonlinear component of a test point  $\mathbf{x}_l \in \mathbb{R}^m$  is obtained as in Eq. (2.30), but changing  $\mathbf{K}_l$  by  $\check{\mathbf{K}}_l = [\langle \phi(\mathbf{x}_l), \phi(\mathbf{x}_i) \rangle_{i \in \text{Ind}(y=j)}]_{j=1, \dots, c}^{i=1, \dots, n_j}$ , where  $\text{Ind}(y=j)$  returns the set of sample indices in the  $j^{th}$  class.

### 2.3.3 Projection Pursuit

Linear transformations have been widely used as feature extraction methods to overcome problems related to high-dimensional spaces. Recently a generalized linear projection method, known as (exploratory) projection pursuit (PP), has been applied as a preprocessing stage in high dimensional classification tasks dealing with face recognition [54] and hyperspectral image analysis [26, 27, 55]. PP was defined as a technique for exploratory analysis of large multivariate datasets, aimed at unsupervised dimensionality reduction and feature extraction [56]. Projection pursuit tackles the curse of dimensionality by means of projecting the whole dataset onto a low dimensional embedding, which should retain the same amount of useful information as the original dataset, and help to visualize the underlying data structure. The optimal projection coefficient set is found by looking at the data degree of interestingness, which is assessed by a predefined index function known as projection index.

The first work on projection pursuit was in 1969 by Kruskal [57], but was Friedman and Tukey [56] who introduced the term. Several developments were inspired after their work, extending the idea of projection pursuit (PP) to other contexts, like projection pursuit regression [58] and projection pursuit density estimation [59]. But it was until Jones and Sibson [60] established a framework based on information theory principles, when this methodology drew the attention of researches. In the following a description of the optimisation problem pose by PP and the characteristics of representative projection indices is provided.

#### 2.3.3.1 The optimisation problem

Unlike PCA or LFD, PP does not count with a close solution. Instead, a suitable optimisation method must be applied to solve the optimisation problem pose as follows. Given the learning data defined in Section 2.3, and a projection pursuit index  $\mathfrak{S}$  that measure the degree of interestingness in the projected data defined as in Eq. (2.17), we look for a projection matrix  $\mathbf{P}^*$  that maximises the criterion

$$\mathbf{P}^* = \underset{\substack{\mathbf{P} \in \mathbb{R}^{m \times b} \\ \mathbf{P}^T \mathbf{P} = \mathbf{I}_b}}{\text{argmax}} \mathfrak{S}(\mathbf{X}\mathbf{P}) \quad (2.34)$$

#### 2.3.3.2 Current projection indices

The selection of an adequate projection index plays an important part in PP, since it defines the properties one wants to highlight in the projected subspace. Generally

Table 2.1: PDF-Based Indices.

Index	Characteristics
Friedman-Tukey [60] $\mathfrak{S}_1 \equiv \int \hat{f}^2 dz$	Minimised by a parabolic density function. Easy to compute. Not sensitive to outliers.
Fisher information criterion [61] $\mathfrak{S}_2 \equiv \int (\hat{f}')^2 / \hat{f} dz$	Represents a non-entropy based index, which is uniquely optimised by a normal pdf. Requires an approximation of the pdf derivative.
Jones-Sibson [60] $\mathfrak{S}_3 \equiv \int -\hat{f} \log(\hat{f}) dz$	It is uniquely optimised by the normal pdf. Provides a natural location for the origin. It highly depends on selected pdf estimation method.
Information divergence [27] $\mathfrak{S}_4 \equiv \int \hat{f} \log(\hat{f}/\phi) dz + \int \phi \log(\phi/\hat{f}) dz$	Allows the definition of uninteresting projections by means of their pdf. A reference pdf has to be computed.

Table 2.2: Moment-Based Indices.

Index	Characteristics
Skewness [26] $\mathfrak{S}_5 \equiv \frac{m_3(\mathbf{z})}{m_2^{3/2}(\mathbf{z})}$	Interesting projections will exhibit negative skewness. Very sensitive to outliers.
Kurtosis [26] $\mathfrak{S}_6 \equiv \frac{m_4(\mathbf{z})}{m_2^2(\mathbf{z})}$	Less prone to outlying projections. Interesting projections will exhibit uniform scatter. Supports ICA.
PCA $\mathfrak{S}_7 \equiv m_2(\mathbf{z})$	Together with the residual subspace technique provides an estimation of PCA as described in [49].
Moments linear combination [60] $\mathfrak{S}_8 \equiv \frac{1}{12} \left( m_3^2(\mathbf{z}) + \frac{1}{4} m_4^3(\mathbf{z}) \right)$	Provides an approximation to entropy. The balance between the forth and third moments may vary.

Table 2.3: Class-Information-Based Indices.

Index	Characteristics
$L_r$ -norm [62] $\mathfrak{S}_9 \equiv \left( \frac{\sum_{j=1}^c (\mu_j - \mu)^r}{\sum_{j=1}^c \sum_{i=1}^{n_j} (z_i - \mu_j)^r} \right)$	LDA is a special case for $r = 2$ . Presents smoother versions of LDA for $r$ equals to multiples of two. Sensitive to outliers as the parameter $r$ increases. In [62] it was optimised using simulated annealing. In the equation $n_j$ is the number of instances in class $j$ , $\mu_j$ is the mean of class $j$ , and $\mu$ is the global mean.
Bharracharya distance [63] $\mathfrak{S}_{10} \equiv \frac{1}{4} \frac{(\mu_i - \mu_j)^2}{\sigma_i^2 + \sigma_j^2} + \frac{1}{2} \log \frac{\sigma_i^2 + \sigma_j^2}{2\sigma_i\sigma_j}$	It is based on first and second order statistics, and is related with the Bayes-classification accuracy. When dealing with $c > 2$ classes, $c(c - 1)/2$ computations are needed.

speaking, PP indices can be classified into three categories: pdf-based indices, moment-based indices and class-information-based indices.

Among the most popular pdf-based indices are order-1 entropy [60], Fisher information criterion [61],  $L^2$ -metric [64], Hermitian index [65], and information divergence [27]. All this indices look for departure from Gaussian distributions, since they are considered uninteresting. Their design facilitate the discovery of clusters, typically characterised by multimodal distributions. A drawback in the optimisation of pdf-based indices is that their robustness against outlying projections highly depends on the selected pdf estimation technique.

Moment-based indices [26], like kurtosis, skewness, and the linear combination of central sample moments, have being used to approximate entropy and avoid probability density function (pdf) estimation. They have shown to be particularly useful in unsupervised detection of small man-made targets distributed on an unknown image scene. This ability to detect outlying projections is as well a disadvantage for classification tasks, since the presence of outliers may hide projections that allow a clear cut between classes.

Indices computed with class information consider data structures of different classes as interesting, for which the frequent choice is linear discriminant analysis (LDA) based on different computations of distance, such as  $L_r$ -norm [62] and Bhattacharyya distance between two classes [63]. The  $L_r$  index offers a tradeoff between discrimination ability and robustness against outliers as its parameter  $r$  is varied. Increasing the numeric value of  $r$ , increases the ability of the index to detect outlying projections, while decreasing

it improves its resilience against outliers. Like any LDA based technique, the  $L_r$ -norm index suffers from the small sample size problem; this problem has been recently discussed in [66] where regularisation is applied to the index proposed in [62] and the PP components are computed by means of simulating annealing.

### 2.3.3.3 Extension to multiple dimensions

**Parallel projection pursuit.** Although the classic formulation for PP optimises only a one-dimensional projection and then expands its definition to more than one projection, the final optimal projection matrix  $\mathbf{P}^*$  in Eq. (2.34) has been presented without further detail. However, it is needed to clarify that there are two different approaches in PP to build  $\mathbf{P}^*$ . The first approach is known as parallel projection pursuit (PPP) [56] and works by jointly optimising every component in the projection matrix by computational expensive methods. It uses symmetric orthogonalisation as described in Section 2.3.1.3, but on Step 3 of the algorithm the optimisation described by Eq. (2.34) is performed for  $b = 1$ .

**Sequential projection pursuit.** The second method is known as sequential projection pursuit (SPP), it finds the best one-dimensional projection  $\mathbf{p}_j^*$  corresponding to the  $j^{\text{th}}$  column of  $\mathbf{P}^*$ , and then removes the contribution of such projection from the original feature space [67] by projecting it onto its residual subspace. Hence, the  $(j + 1)^{\text{th}}$  projection vector is given by the solution to

$$\mathbf{p}_{i+1} = \underset{\mathbf{p} \in \mathbb{R}^m}{\operatorname{argmax}} \mathfrak{S}(\mathbf{X}\mathbf{S}_j^\perp \mathbf{p}), \quad (2.35)$$

where  $\mathbf{S}_j^\perp = \mathbf{I}_m - \mathbf{P}_j^* \mathbf{P}_j^{*T}$  is the orthogonal complement of  $\mathbf{P}_j^* = [\mathbf{p}_1^*, \mathbf{p}_2^*, \dots, \mathbf{p}_j^*]$ , which is the optimal projection matrix for the first  $j$  projection vectors. Although PPP gives all the projections components in two steps of the algorithm, the symmetric orthogonalisation may disrupt the optimal solution for the cost function and not always converges. Thus SPP was adopted as the preferred method to build  $\mathbf{P}^*$ .

**Stopping criterion for SPP.** A recently proposed stopping criterion which relies on Bayesian model selection [21] was selected to automatically determine the number of projections in the SPP procedure. It is based on the fact that the remaining structure of the residual search space is decreased as the number of components increases. The stopping criterion is defined as

$$B = \left( 2^{n \times [\mathfrak{S}(\mathbf{X} \cdot \mathbf{p}_{i-1}^*) - \mathfrak{S}(\mathbf{X} \cdot \mathbf{p}_i^*)]} + 1 \right)^{-1} \quad (2.36)$$

and includes the  $i^{\text{th}}$  projection component  $\mathbf{p}_i^*$  if  $B$  is bigger than a predefined threshold  $\delta$ , otherwise SPP stops with  $i-1$  projection vectors in  $\mathbf{P}^*$ .

### 2.3.4 Spectral Embedding Methods

Although nonlinear feature extraction in a kernel-induced feature space is successfully performed by the previously described algorithms, valuable information for the classification task may be lost together with the discarded eigenvectors when performing dimensionality reduction. Additionally, recent evidence suggests that high-dimensional spaces may allow a nonlinear embedding of data that originally could be lying in a lower-dimensional manifold [68]. Under this assumption, a number of unsupervised spectral embedding (USE) methods have been proposed [69–71], along with their linear out-of-sample extension [13, 72–74]. They preserve certain characteristics of the original high-dimensional space, such as aggregate pairwise proximity information based on local neighbourhood graphs. Nevertheless, in a supervised classification task, neighbouring points near the class boundaries may get projected to the wrong class, damaging classification performance. Several supervised spectral embedding (SSE) alternatives have been proposed to alleviate this problem. Those closely related to LFD [75–77] make use of the between- and within-class information to restrict the embeddings, whereas a second class of algorithms modify the proximity definition as to consider the label information [28, 78–80].

The similarity in the optimisation problem pose by embedding methods has inspired several works to formulate an unification framework [13, 77, 81]. In this thesis, the dual formulation of the template proposed in [77] is adopted since it allow us a wider representation of embedding methods. Such dual formulation can be expressed as

$$\max_{\substack{\mathbf{Z} \in \mathbb{R}^{n \times b}, \\ \mathbf{Z}^T \mathbf{B} \mathbf{Z} = \mathbf{I}_b}} \text{tr} \left[ \mathbf{Z}^T \mathbf{A} \mathbf{Z} \right], \quad (2.37)$$

where  $\mathbf{A} \in \mathbb{R}^{n \times n}$  can be viewed as an arbitrary weight matrix,  $\mathbf{B} \in \mathbb{R}^{n \times n}$  as a scale or label information constraint matrix, and  $\mathbf{I}_b$  is a  $b \times b$  identity matrix. Thus the optimal embedding is given by the eigenvectors of  $\mathbf{B}^{-1} \mathbf{A}$  corresponding to its  $b$  largest eigenvalues. This template allow to express several embedding methods by changing the structure of  $\mathbf{A}$  and  $\mathbf{B}$ . In the following, a brief description of the existing unsupervised and supervised algorithms is provided under the foregoing unification framework.

To accommodate an out-of-sample extension, the definition of linear projection in Eq. (2.17) is used to rewrite the template in Eq. (2.37) as

$$\max_{\substack{\mathbf{P} \in \mathbb{R}^{m \times b} \\ \mathbf{P}^T \mathbf{B}_p \mathbf{P} = \mathbf{I}_b}} \text{tr} \left[ \mathbf{P}^T \mathbf{A}_p \mathbf{P} \right], \quad (2.38)$$

where  $\mathbf{A}_p = \mathbf{X}^T \mathbf{A} \mathbf{X}$  and  $\mathbf{B}_p$  are the objective and constraint matrices, respectively. This modification allows the projection matrix  $\mathbf{P}$  to be computed directly, and through Eq. (2.17) a test point can be straightforward projected into the embedding.

### 2.3.4.1 Unsupervised methods

**Isomap** [69] uses an estimation of geodesic distance among neighbouring points to build a similarity matrix  $\mathbf{W}$ , then classical multidimensional scaling (MDS) [82] is applied to transform  $\mathbf{W}$  into a Gram matrix  $\mathbf{A}$  by means of the double centering transformation

$$\mathbf{A} = -\frac{1}{2}(\mathbf{I}_n - \mathbb{1}_n) \mathbf{W} (\mathbf{I}_n - \mathbb{1}_n). \quad (2.39)$$

Thus, the optimal embedding is given by the solution to Eq. (2.37) with  $\mathbf{B} = \mathbf{I}_n$ .

**Local linear embedding** (LLE) [70] preserves the structure within a neighbourhood, regarding the manifold as a set of intersecting patches. Each patch is assumed to be linear and its structure is represented by a set of local predictive weights such that  $\sum_j w_{ij} = 1$ ,  $w_{ii} = 0$ ,  $w_{ij} = 0$  if  $\mathbf{x}_j$  is not within the neighbourhood of  $\mathbf{x}_i$  and  $\|\mathbf{x}_i - \sum_j w_{ij} \mathbf{x}_j\|^2$  is minimised. Thus, for each point  $\mathbf{x}_i$ , its corresponding local predictive weights are computed as

$$\mathbf{w}_i = \frac{\mathbf{G}^{-1} \mathbf{1}_n}{\mathbf{1}_n^T \mathbf{G}^{-1} \mathbf{1}_n}, \quad (2.40)$$

where  $\mathbf{G}$  is a local Gram matrix which depict the covariance in the neighbourhood  $\eta_i$  of  $x_i$ , with  $g_{ij} = (\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)$ , if  $\mathbf{x}_j \in \eta_i$ , or  $g_{ij} = 0$  otherwise. To obtain the embedding, each point in the lower-dimensional space is represented as well by a weighted sum of its neighbours and the error  $\sum_i \|\mathbf{z}_i - \sum_j w_{ij} \mathbf{z}_j\|^2$  is minimised with respect to  $\mathbf{z}_1, \dots, \mathbf{z}_n \in \mathbb{R}^b$  using the local predictive weights  $\mathbf{W}$  previously computed. This quadratic form can be rewritten in terms of inner products as

$$\sum_{ij} \left( \delta_{ij} - w_{ij} - w_{ji} + \sum_r w_{ri} w_{rj} \right) (\mathbf{z}_i^T \mathbf{z}_j). \quad (2.41)$$

This criterion can be rewritten in order to meet the template proposed in Eq. (2.37), such that the required matrices  $\mathbf{A}$  and  $\mathbf{B}$  are expressed as

$$\mathbf{A} = \mathbf{W} + \mathbf{W}^T - \mathbf{W}^T \mathbf{W} \text{ and } \mathbf{B} = \mathbf{I}_n \quad (2.42)$$

**Laplacian eigenmaps** (LE) [71] represents the dataset as a connected graph and uses its Laplacian to compute a low-dimensional embedding. First a matrix  $\mathbf{W}$  of edge weights is built according to one of two criteria: 1)  $w_{ij} = 1$  if  $\mathbf{x}_j$  is one of the  $k$ -nearest neighbours of  $\mathbf{x}_i$ , and 0 otherwise. 2)  $w_{ij} = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\tau}\right)$  if  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are connected nodes, 0 otherwise. Then the Laplacian matrix given by  $\mathbf{L} = \mathbf{D}(\mathbf{W}) - \mathbf{W}$  is computed, where  $\mathbf{D} \equiv \mathbf{D}(\mathbf{W}) = \text{diag}(\mathbf{W} \mathbf{1}_n)$  is a diagonal matrix indicating the degree of each node. Under the assumption that connected points remain as close as possible in the new subspace, the solution to the following optimisation problem gives the low-dimensional embedding

$$\min_{\substack{\mathbf{Z} \in \mathbb{R}^{n \times b}, \\ \mathbf{Z}^T \mathbf{D} \mathbf{Z}}} \text{tr} \left[ \mathbf{Z}^T \mathbf{L} \mathbf{Z} \right], \quad (2.43)$$



which can be formulated in terms of Eq. (2.37) with matrices  $\mathbf{A} = \mathbf{W}$  and  $\mathbf{B} = \mathbf{D}$ .

**Locality preserving projections** (LPP) [13] is a direct linearization of LE, where the connected graph representation and the definition of the Laplacian matrix are kept. Thus, LPP is formulated under Eq. (2.38) using matrices  $\mathbf{A} = \mathbf{W} - \mathbf{D}$  and  $\mathbf{B}_p = \mathbf{X}^T \mathbf{D} \mathbf{X}$ . It is worth to notice that the extracted features using LPP are not longer orthogonal.

**Orthogonal neighbourhood preserving projections** (ONPP) [72] seeks for an orthogonal mapping that best preserves local connectivity among neighbours in the graph. Similar to LLE, ONPP builds a weighting matrix  $\mathbf{W}$  according to Eq. (2.40) to describe the local structure to be preserved. It also imposes the same reconstruction error constraint over each point on the reduced space, but the orthogonality constraint is imposed over the projection matrix rather than over the projected points. Hence, ONPP is defined under the proposed template by the matrices  $\mathbf{A} = \mathbf{W} + \mathbf{W}^T - \mathbf{W}^T \mathbf{W} - \mathbf{I}_n$  and  $\mathbf{B}_p = \mathbf{I}_m$

#### 2.3.4.2 Supervised methods

**Local Fisher discriminant analysis** (LFDA) [76] combines the ideas behind LPP and FDA by incorporating local information in the definition of the within-class and between-class scatters. The local counterparts of Eq. (2.22), as well as the original matrices, can be written as weighted sums of pairwise distances as follows

$$\tilde{\mathbf{S}}_W = \frac{1}{2} \sum_{i,j=1}^n \hat{h}_{ij} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T \text{ and } \tilde{\mathbf{S}}_B = \frac{1}{2} \sum_{i,j=1}^n h_{ij} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T, \quad (2.44)$$

where

$$\hat{h}_{ij} = \begin{cases} \mathbf{w}_{ij}/n_a & \text{if } y_i = y_j = a, \\ 0 & \text{otherwise.} \end{cases} \text{ and } h_{ij} = \begin{cases} \mathbf{w}_{ij}(1/n - 1/n_a) & \text{if } y_i = y_j = a, \\ 1/n & \text{otherwise.} \end{cases} \quad (2.45)$$

Note the use of the affinity matrix  $\mathbf{W}$  to weight the pairwise distance between samples in the same class. Such matrix can be computed using several pairwise functions such as the Gaussian kernel, the cosine norm, Person's, Spearman's or Kendall's correlation coefficients, or any user-defined similarity function. If the weights  $\mathbf{w}_{ij}$  are removed from Eq. (2.45), the definitions in Eq. (2.44) and (2.22) are equivalent. From Eq. (2.44) it can be shown that each scatter matrix can be expressed as the Laplacian matrix of a connected graph with weights  $\mathbf{H}$  and  $\hat{\mathbf{H}}$ , respectively. Thus, the optimal, discriminative, projection basis can be computed using the proposed template with matrices  $\mathbf{A} = \mathbf{H} - \mathbf{D}(\mathbf{H})$  and  $\mathbf{B}_p = \mathbf{X}^T (\mathbf{D}(\hat{\mathbf{H}}) - \hat{\mathbf{H}}) \mathbf{X}$ .

**Maximum margin criterion** (MMC) [75] uses the summation of all pair inter-class margins as feature extraction criterion, which is defined as

$$J = \frac{1}{2} \sum_{i,j=1}^c p_i p_j \left( d(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) - \text{tr}(\boldsymbol{\Sigma}_i) - \text{tr}(\boldsymbol{\Sigma}_j) \right), \quad (2.46)$$

where  $d(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) = (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$  is the distance between class mean vectors. It can be shown such criterion reduces to

$$J = \text{tr}(\mathbf{S}_B - \mathbf{S}_W), \quad (2.47)$$

thus the optimal projection basis can be computed through Eq. (2.38) with matrices  $\mathbf{A}_p = \mathbf{S}_B - \mathbf{S}_W$  and  $\mathbf{B}_p = \mathbf{I}_m$ . MMC overcomes the small sample size problem by not needing to compute the inverse of the within-class scatter matrix.

**Discriminant neighbourhood embedding** (DNE) [83] incorporates class information into the affinity matrix definition under the framework pose by LPP. This refinement in the algorithm result in a criterion based on a not positive-semidefinite matrix  $\mathbf{D}(\mathbf{W}) - \mathbf{W}$ , where

$$w_{ij} = \begin{cases} +1 & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are intra-class kNNs,} \\ -1 & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are inter-class kNNs,} \\ 0 & \text{otherwise.} \end{cases} \quad (2.48)$$

is the modified affinity matrix with label information. A sample  $\mathbf{x}_k$  is considered the inter-class nearest neighbour of  $\mathbf{x}_i$  if  $|\mathbf{x}_i - \mathbf{x}_k| < |\mathbf{x}_i - \mathbf{x}_j|, \forall y_j : y_j = y_k \wedge y_j \neq y_i$ . In terms of the adopted template, DNE can be expressed in terms of matrices  $\mathbf{A} = \mathbf{W} - \mathbf{D}(\mathbf{W})$  and  $\mathbf{B}_p = \mathbf{I}_m$ .

**Supervised orthogonal neighbourhood preserving projections** (SONPP) [72] modifies ONPP to take into account only intra-class examples as connected components in the graph, eliminating the need to define the number of nearest neighbours  $k$  involved in the optimal reconstruction process. In consequence, after sorting its rows and columns using the class label of each sample, the predictive weights take the form of a block diagonal matrix  $\widetilde{\mathbf{W}} = \text{diag}(\mathbf{W}_1, \mathbf{W}_2, \dots, \mathbf{W}_c)$ , where each block  $\mathbf{W}_j = [\mathbf{w}_i]_{i=1, \dots, n_j}$  is computed as in Eq. (2.40) with samples of the  $j^{\text{th}}$  class. Being a direct modification of ONPP, the optimal projection matrix in SONPP can be computed using the selected template with matrices  $\mathbf{A} = \widetilde{\mathbf{W}} + \widetilde{\mathbf{W}}^T - \widetilde{\mathbf{W}}^T \widetilde{\mathbf{W}} - \mathbf{I}_n$  and  $\mathbf{B}_p = \mathbf{I}_m$ .

**Repulsion Laplaceans** [78] established the concept of repulsion graph, defined as a undirected graph where a node  $i$  is connected only to its  $k$ -nearest enemies, which are the nodes corresponding to the  $k$ -nearest samples not in the same class as  $\mathbf{x}_i$ . The weighting matrix of the repulsion graph  $\mathbf{W}^r$  is built using the heat kernel as in LE,

the regression coefficients as in LLE, or an alternative weight function proposed in [78]. The aim of the repulsion graph is to create repulsion forces that will separate points from different classes that were originally close. Two methods were derived in [78] from the introduction of the repulsion graph to classic graph embedding methods. The first one, named OLPP-R, is a direct modification to LPP formulation, where the laplacian matrix in Eq. (2.43) is replaced by a linear combination of a repulsion laplacian  $\mathbf{L}^r = \mathbf{D}(\mathbf{W}^r) - \mathbf{W}^r$  and a class laplacian  $\mathbf{L}^s = \mathbf{D}(\mathbf{W}^s) - \mathbf{W}^s$ , where  $w_{ij}^s = 1/n_a$  if  $y_i = y_j = a$ , 0 otherwise. The final laplacian for OLPP-R is  $\mathbf{L} = \mathbf{L}^s - \beta\mathbf{L}^r$ , where  $\beta > 0$  is a user-defined parameter. OLPP-R can be expressed under Eq. (2.38) using matrices  $\mathbf{A} = \mathbf{H} - \mathbf{D}(\mathbf{H})$  and  $\mathbf{B}_p = \mathbf{I}_m$ , where

$$h_{ij} = \begin{cases} \frac{1}{n_a} & \text{if } y_i = y_j = a, \\ -\beta w_{ij}^r & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are inter-class kNNs,} \\ 0 & \text{otherwise.} \end{cases} \quad (2.49)$$

**Discriminative orthogonal neighbourhood preserving projections (DONPP)** [84] keeps the intra-class structure by minimising the reconstruction error in the low-dimensional embedding using the predictive weights trained as in Eq. (2.40). Additionally, DONPP maximises the distances between each sample and its inter-class kNNs to keep a neat separation among classes. This process is modelled as a repulsion graph [78] where only inter-class kNNs are made adjacent. Using a binary adjacency matrix

$$w'_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are inter-class kNNs} \\ 0 & \text{otherwise} \end{cases}$$

to incorporate inter-class information to the optimisation problem stated by SONPP, DONPP is formulated under Eq. (2.38) with the aid of matrices  $\mathbf{A} = \widetilde{\mathbf{W}} + \widetilde{\mathbf{W}}^T - \widetilde{\mathbf{W}}^T \widetilde{\mathbf{W}} - \mathbf{I}_n + \beta(\mathbf{D}(\mathbf{W}') - \mathbf{W}')$  and  $\mathbf{B}_p = \mathbf{I}_m$ , where  $\beta$  is a constant controlling the contribution of the penalty term.

### 2.3.5 Feature Extraction for High-Dimensional Spaces

The problem of extracting compact, highly informative features to improve computational efficiency for the classification task of high-dimensional data has received a lot of attention in the last couple of years. The difficulty in directly applying machine learning algorithms to high-dimensional data lie on the curse of dimensionality [85]. Usually, feature extraction methods are employed as a preprocessing stage to tackle this problem, mapping the original input space  $\mathcal{M}$  to a lower-dimensional one  $\mathcal{B}$ . According to the mapping structure, feature extraction methods can be categorised as linear or nonlinear.

Linear feature extraction techniques improve the discriminatory characteristics of the data, translating the original samples to a lower-dimensional representation using

a linear projection matrix. The commonly used linear methods include PCA [47], ICA [49], and LFD [42]. PCA can dramatically reduce the dimensions of the input space by providing an orthonormal basis where the total scatter of the projected samples is maximised, nevertheless it can discard useful information for classification. LFD produces highly discriminatory features by directly maximising an explicit measure of class separability (i.e., Rayleigh quotient). One of the drawbacks of LFD is present when the dimensionality of the data exceeds the sample size. Under this situation, known as the singularity or undersampled problem, all the scatter matrices in LFD become singular and classical LFD cannot be applied. Several techniques have been proposed to alleviate this problem [86–91]. Contrary to PCA and LFD, ICA generates non-orthogonal, highly independent features by iteratively optimising a measure of nongaussianity. Thus, impeding its use when the original feature space presents normal distributions.

Linear projection techniques have been successfully applied to a broad diversity of tasks, however in many real-world problems, a linear mapping is not enough to model the underlying nonlinear data structure. Different research has been developed for nonlinear feature extraction, and can be roughly classified into spectral embedding methods and kernel-induced feature space methods. Spectral embedding methods [13, 72–74] are based on the assumption that the underlying structure of a high-dimensional dataset can be embedded into a low-dimensional subspace, where a measure of similarity between neighbours must be preserved. They have shown good performance on artificial datasets. A popular method in this category is LPP [74], it works by providing a low-dimensional embedding retaining the distance among samples in a local neighbourhood. Kernel-induced feature space methods make use of the kernel trick [52] to project the original input space to a non-observable feature space, accessible only through its dot product given by a kernel function evaluated at those points. The core idea is that the projection to the non-observable feature space will unfold any nonlinearity and will make possible to apply the linear projection techniques in the non-observable space. Under this assumption, extensions of the most popular linear feature extraction methods have been proposed such as KPCA [51], and KFD [53, 92]. Similarly to its linear counterpart, KFD presents the singularity problem which have been solved by means of regularisation [93, 94].

## Chapter 3

# Optimisation Techniques in Machine Learning

Optimisation, also known as mathematical programming (MP) is a broad subfield in operations research that deals with the selection of a best element from some set of available alternatives. Since the cost of a potential solution can be expressed as a function of some decision variables, optimisation can be formulated as well as the process of finding the minimum of the cost function, within a constraint domain. ML and MP present an intrinsic relationship that has been studied and exposed in [95, 96]. Understanding the common grounds between ML and MP has allowed improvement and development of existing and new learning models, respectively, based on popular optimisation methods. In ML, optimisation methods are used in three different problems namely classification, parameter estimation and model selection. The last two problems often are presented as nested optimisation problems, where several instances of parameter estimation have to be solved in the process of model selection.

Formally speaking, an optimisation problem can be stated as

$$\begin{aligned} \text{Find } \mathbf{x} = [x_1, x_2, \dots, x_n]^T \text{ which minimises } & f(\mathbf{x}) \\ \text{s.t. } & g_j(\mathbf{x}) \leq 0, j = 1, \dots, m \\ & l_i(\mathbf{x}) = 0, i = 1, \dots, p \end{aligned} \quad (3.1)$$

where the minimum of the objective function  $f(\mathbf{x})$  is reached at some values of the  $n$ -dimensional design vector  $\mathbf{x}$  that meet the equality constraints  $l_i(\mathbf{x})$  and inequality constraints  $g_j(\mathbf{x})$ . From the above formulation, MP problems can be categorised into two wide and general classes based on the absence or existence of both type of constraints, namely *unconstrained* or *constrained problems*. A more convenient and commonly used classification is based on the structure of the objective function and constraint equations, grouping MP problems into nonlinear, linear, geometric and quadratic programming problems. For every of the aforementioned categories exist specific optimisation algorithms for their efficient solution. In the remaining of this section a brief description of the relevant optimisation algorithms used in this work is provided.

## 3.1 Standard Optimisation Techniques

### 3.1.1 Linear Programming

Linear programming [97–99] is applicable to optimisation problems in which the objective function and the constraints are formulated as linear combinations of the decision variables. Thus, the general structure presented in Eq. (3.1) takes the matrix form

$$\begin{aligned} \min f(\mathbf{x}) &= \mathbf{c}^T \mathbf{x} \\ \text{s.t. } \tilde{\mathbf{A}}\mathbf{x} &= \mathbf{b} \\ \mathbf{x} &\geq \mathbf{0} \end{aligned} \tag{3.2}$$

where  $\tilde{\mathbf{A}} = [a_{ij}]_{j=1, \dots, n}^{i=1, \dots, m}$  is a mixing matrix,  $\mathbf{b} = [b_i]_{i=1, \dots, m}^T$  is a bias column vector, and  $\mathbf{c} = [c_i]_{i=1, \dots, n}^T$  is a weighting column vector. It is assumed that  $m < n$  such that the linear system  $\tilde{\mathbf{A}}\mathbf{x} = \mathbf{b}$  has an infinite number of solutions from which the one minimising  $f(\mathbf{x})$  is selected, or it has no solution at all. The other two cases when  $m > n$  and  $m = n$  are of no interest since either the problem is overspecified or has a unique solution. When the optimisation problem is properly formulated, the feasible region (region defined by the intersection of the constraints domain) is a convex polyhedron, and the optimum value occurs at an extreme point or vertex of the feasible region.

Since the possible number of potential solutions on the feasible region increases with the number of design variables, searching for the optimal solution becomes a cumbersome task. An iterative method known as *simplex algorithm* provides an efficient searching strategy consisting of two phases. Phase I constructs an auxiliary problem by the introduction of artificial variables, then its optimal solution is found by a sequence of pivotal operations. The optimal solution to the auxiliary problem coincides with a basic feasible solution of the original problem. If such solution is not optimal, Phase II delivers a neighbouring feasible solution with a lower or equal value of  $f(\mathbf{x})$  by means of a second sequence of pivotal operations. This step is repeated until an optimal solution is found.

### 3.1.2 Quadratic Programming

A quadratic programming problem [100, 101] is the simplest nonlinear programming case, with a quadratic objective functions and linear constraints. Different from more complex nonlinear programming cases, where the derivatives of the objective functions can not be obtained, a quadratic programming problem can be solved by transforming the problem into a linear programming problem through the use of the Lagrange multipliers and the Kuhn-Tucker conditions [102]. The formulation of a quadratic pro-

gramming with linear constrains can be stated as follows

$$\begin{aligned} \min f(\mathbf{x}) &= \mathbf{c}^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T \mathbf{U} \mathbf{x} \\ \text{s.t. } \tilde{\mathbf{A}} \mathbf{x} &\leq \mathbf{b} \\ \mathbf{x} &\geq \mathbf{0} \end{aligned} \quad (3.3)$$

where the term  $\mathbf{x}^T \mathbf{U} \mathbf{x} / 2$  is the quadratic part of the objective function with  $\mathbf{U} = [u_{ij}]_{j,i=1,2,\dots,n}$  being a symmetric positive-definite matrix. By using surplus variables  $t_j^2$ ,  $j = 1, 2, \dots, n$  and slack variables  $s_i^2$ ,  $i = 1, 2, \dots, m$ , the above optimisation problem can be pose in standard form

$$\begin{aligned} \min f(x) &= \mathbf{c}^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T \mathbf{U} \mathbf{x} \\ \text{s.t. } \tilde{\mathbf{A}}_i^T \mathbf{x} + s_i^2 &= b_i, \quad i = 1, 2, \dots, m \\ -x_j + t_j^2 &= 0, \quad j = 1, 2, \dots, n \end{aligned} \quad (3.4)$$

where  $\tilde{\mathbf{A}}_i = [a_{i1}, a_{i2}, \dots, a_{in}]^T$  is the  $i^{\text{th}}$  column of the mixing matrix. Using Lagrange multipliers  $\lambda_i$  and  $\theta_j$ , the Kuhn-Tucker conditions for the stationariness of the Lagrange function for the optimisation problem are given as

$$c_j - \theta_j + \sum_{i=1}^n u_{ij} x_j + \sum_{i=1}^m a_{ij} \lambda_i = 0, \quad j = 1, 2, \dots, n \quad (3.5a)$$

$$\tilde{\mathbf{A}}_i^T \mathbf{x} + Y_i = b_i, \quad i = 1, 2, \dots, m \quad (3.5b)$$

$$\mathbf{x} \geq 0, \mathbf{Y} \geq 0, \boldsymbol{\lambda} \geq 0, \boldsymbol{\theta} \geq 0 \quad (3.5c)$$

$$\lambda_i Y_i = 0, \quad i = 1, 2, \dots, m \quad (3.5d)$$

$$\theta_j x_j = 0, \quad j = 1, 2, \dots, n \quad (3.5e)$$

where  $\mathbf{Y} = [Y_i]$ ,  $Y_i = s_i^2 \geq 0$ ,  $\boldsymbol{\lambda} = [\lambda_i]$ , and  $\boldsymbol{\theta} = [\theta_j]$ , for  $i = 1, 2, \dots, m$ , and  $j = 1, 2, \dots, n$ . In the above formulation, all but the last two equations are linear functions of the variables  $x_j$ ,  $Y_i$ ,  $\lambda_i$  and  $\theta_j$ . Hence, the optimal solution for Eq. (3.4) is given by the feasible solution for the  $2(n + m)$  simultaneous system described by Eqs. (3.5a)-(3.5e). Although a basic feasible solution can be a local minimum, global optimality is guaranteed as convexity of  $f(\mathbf{x})$  in Eq. (3.4) is granted by the positive-definitive matrix  $\mathbf{U}$ . The required basic feasible solution can be obtained by Phase I of the simplex method on the system described by Eqs. (3.5a) and (3.5b), reformulated as the linear problem

$$\begin{aligned} \min F &= \sum_{j=1}^n z_j \\ \text{s.t. } c_j - \theta_j + \sum_{i=1}^n u_{ij} x_j + \sum_{i=1}^m a_{ij} \lambda_i + z_j &= 0, \quad j=1, 2, \dots, n \\ \tilde{\mathbf{A}}_i^T \mathbf{x} + Y_i &= b_i, \quad i=1, 2, \dots, m \\ \mathbf{x} \geq 0, \mathbf{Y} \geq 0, \boldsymbol{\lambda} \geq 0, \boldsymbol{\theta} &\geq 0 \end{aligned} \quad (3.6)$$

where  $z_j$  are artificial variables. When performing the sequence of pivoting operations to reach the feasible solution, one must make sure the additional constraints Eqs. (3.5d) and (3.5e) are satisfied by keeping into the basic solution either  $Y_i$  or  $\lambda_i$ , but not both. Similar care has to be taken for variables  $\theta_j$  and  $x_j$ .

### 3.1.3 Nonlinear Unconstrained Programming

A nonlinear programming problem (NLP) is the most generic type of optimisation problems, where any of the functions among the objective and constrains is nonlinear. Although nonlinear unconstrained problems are rare, their study is important as it provides the basis for most of the powerful methods for solving constraint nonlinear problems [103]. Given an unconstraint minimisation problem with objective function  $f(\mathbf{x})$ , a local minimum exist at point  $\mathbf{x}^*$  if the necessary conditions

$$\frac{\partial f}{\partial x_i}(\mathbf{x} = \mathbf{x}^*) = 0, \quad i = 1, 2, \dots, n \quad (3.7)$$

are satisfied, along with the sufficient condition restricting the Hessian

$$\mathbf{J}_{\mathbf{x}^*} = [\mathbf{J}]_{\mathbf{x}^*} = \left[ \frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}^*) \right] \quad (3.8)$$

to be positive semidefinite. When the objective function is not differentiable in every point of its domain, the above equations cannot be used to find a minimum, and analytical methods can not be used to derive a close form solution. However, numerical methods provide suitable approximations to the minimum, under certain conditions [104,105]. They can be divided into tow broad categories as direct search methods and gradient methods. Direct search methods overcome this problem by not requiring the computation of partial derivatives, nevertheless these methods only give satisfactory results for problems of low dimensionality.

#### Gradient Descent Methods

Gradient methods are a powerful alternative for the solution of nonlinear unconstrained minimisation problems. They are based on the concept of steepest descent motion, which states the objective function will decrease at the fastest rate in the opposite direction of the gradient vector  $\nabla f(\mathbf{x}) = [\partial f/\partial x_1, \partial f/\partial x_2, \dots, \partial f/\partial x_n]^T$ . Although theoretical sound, gradient methods are restricted to problems where the objective function is differentiable, but its gradient is either impractical or impossible to derive, or computationally expensive. In such cases, approximations to the gradient at a given point  $\mathbf{x}_k$  are computed using the forward finite-differences formula

$$\left. \frac{\partial f}{\partial x_i} \right|_{\mathbf{x}_k} \simeq \frac{f(\mathbf{x}_k + \Delta x_i \mathbf{u}_i) - f(\mathbf{x}_k)}{\Delta x_i}, \quad (3.9)$$

where a small quantity  $\Delta x_i$  is added in the direction given by the unitary vector  $\mathbf{u}_i$  parallel to the  $i^{th}$  axis. Selection of  $\Delta x_i$  plays an important role in the accuracy of the approximation. A extremely small value in  $\Delta x_i$  may lead to round-off errors, while a large one can cause truncation errors.

Given a starting point  $\mathbf{x}_1$ , gradient methods gradually approach the minimum of  $f(\mathbf{x})$  by moving from point  $\mathbf{x}_i$  to point  $\mathbf{x}_{i+1}$  in the direction  $-\nabla f$ . An iterative sequence of approximations to the minimum can be expressed as



1. Start with an initial position  $\mathbf{x}_1$
2. Find the displacement direction  $\mathbf{S}_i = -\nabla f_i = -\nabla f(\mathbf{x}_i)$
3. Determine the optimal displacement amount  $\varpi_i^*$  in direction  $\mathbf{S}_i$  and compute

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \varpi_i^* \mathbf{S}_i \quad (3.10)$$

4. Stop if any of the following:

$$\left| \frac{f(\mathbf{x}_{i+1}) - f(\mathbf{x}_i)}{f(\mathbf{x}_i)} \right| \leq \varepsilon_1, \quad \text{or} \quad \left| \frac{\partial f}{\partial x_i} \right|_{i=1,2,\dots,n} \leq \varepsilon_2, \quad \text{or} \quad |\mathbf{x}_{i+1} - \mathbf{x}_i| \leq \varepsilon_3,$$

otherwise, go to step 2.

Such iterative algorithm is known as *Cauchy method* [106]. Parameter  $\varpi_i^*$  in step 3 can be found by minimising the rate of change of  $f(\mathbf{x}_{i+1})$  along direction  $\mathbf{S}_i$  with respect to parameter  $\varpi_i$ , given by

$$\frac{df}{d\varpi_i} = \sum_{j=1}^n \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial \varpi_i} = \sum_{j=1}^n \frac{\partial f}{\partial x_j} s_{ij} = \nabla f^T \mathbf{S}_i. \quad (3.11)$$

Thus, the condition  $df/d\varpi_i = 0$  must hold for  $\varpi_i$  to be a minimum, and the optimal displacement amount  $\varpi_i^*$  in direction  $\mathbf{S}_i$  is obtained by finding the roots of the characteristic polynomial  $df(\mathbf{x}_i + \varpi_i \mathbf{S}_i)/d\varpi_i = 0$ .

**Newton's method.** Since the steepest descend direction is a local property, Cauchy method can be suboptimal for most problems. By the incorporation of second derivatives of the objective function, second-order methods improve efficiency. A well known second-order method is *Newton's method* [103–105], which is based on the Taylor's series expansion of  $f(\mathbf{x})$  at point  $\mathbf{x} = \mathbf{x}_i$

$$f(\mathbf{x}) = f(\mathbf{x}_i) + \nabla f_i (\mathbf{x} - \mathbf{x}_i) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_i)^T \mathbf{J}_i (\mathbf{x} - \mathbf{x}_i) \quad (3.12)$$

where  $\mathbf{J}_i = \mathbf{J}_{\mathbf{x}_i}$  and  $\nabla f_i$  are the Hessian matrix and the gradient of  $f(\mathbf{x})$  evaluated at point  $\mathbf{x}_i$ . The minimum of  $f(\mathbf{x})$  is found by setting  $\partial f/\partial x_i = 0$ , thus Eq. (3.12) leads to  $\nabla f = \nabla f_i + \mathbf{J}_i (\mathbf{x} - \mathbf{x}_i) = 0$ . Assuming a nonsingular Hessian, an improved version of Eq. (3.10) can be expressed as

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \mathbf{J}_i^{-1} \nabla f_i. \quad (3.13)$$

To guarantee convergence to a minimum for a non-quadratic objective function, Newton's method is modified by including an optimal displacement amount  $\varpi_i^*$  in direction  $\mathbf{S}_i = -\mathbf{J}_i^{-1} \nabla f_i$ , reformulating Eq. (3.13) to match Eq. (3.10), leading to  $\mathbf{x}_{i+1} = \mathbf{x}_i - \varpi_i^* \mathbf{J}_i^{-1} \nabla f_i = \mathbf{x}_i + \varpi_i^* \mathbf{S}_i$ .

**Quasi-newton methods.** Although the improved Newton's method is a powerful minimisation method, it requires computation, and inversion of the Hessian at every iteration of the steepest descent procedure. When the objective function involves high-dimensional and intricate terms, Newton's method turns to be computationally expensive. A practical alternative to avoid computing the Hessian  $\mathbf{J}_i$  is to provide a suitable approximation by means of matrix  $\mathbf{A}_i$ , similarly  $\mathbf{J}_i^{-1}$  can be approximate by a symmetric and positive definite matrix  $\mathbf{B}_i$ . *Quasi-newton methods* [107] find matrix  $\mathbf{B}_i$  (and therefore matrix  $\mathbf{A}_i = \mathbf{B}_i^{-1}$ ) in an iterative manner as

$$\mathbf{B}_{i+1} = \mathbf{B}_i + \Delta\mathbf{B}_i, \quad (3.14)$$

where  $\Delta\mathbf{B}_i$  can be considered as a correction matrix that is added at each iteration of the Newton's method. According to the rank of the correction term, Quasi-newton methods are classified into rank 1 or rank 2 updates. Rank 2 updates are preferred over rank 1 as they guarantee to keep symmetry and positive-definiteness on matrix  $\mathbf{B}_{i+1}$  at every iteration. Popular rank 2 choices are Davidon-Fletcher-Powell (DFP) [108, 109] and Broydon-Fletcher-Goldfarb-Shanno (BFGS) [109–112] updates. Contrary to DFP formula, where the update is stated in terms of the inverse of the Hessian, BFGS provides a direct approximation to the Hessian by means of matrix  $\mathbf{A}_i$ . Taking as starting point the Newton's method update in Eq. (3.13) at iteration  $i$ , BFGS update consists on the following steps

1. Find the displacement direction  $\mathbf{S}_i$  by solving  $\mathbf{A}_i\mathbf{S}_i = -\nabla f_i$
2. Determine the optimal displacement amount  $\varpi_i^*$  in direction  $\mathbf{S}_i$  and compute  $\mathbf{x}_{i+1} = \mathbf{x}_i + \varpi_i^*\mathbf{S}_i$
3. Stop if  $|\nabla f_{i+1}| \leq \varepsilon$ , where  $\varepsilon$  is a tolerance threshold. Otherwise proceed with next step.
4. Update the Hessian with the formula

$$\mathbf{A}_{i+1} = \mathbf{A}_i + \frac{\mathbf{g}_i\mathbf{g}_i^T}{\mathbf{g}_i^T\mathbf{d}_i} - \frac{\mathbf{A}_i\mathbf{d}_i\mathbf{d}_i^T\mathbf{A}_i}{\mathbf{d}_i^T\mathbf{A}_i\mathbf{d}_i}, \quad (3.15)$$

where vectors  $\mathbf{d}_i = \varpi_i^*\mathbf{S}_i$  and  $\mathbf{g}_i = \nabla f_{i+1} - \nabla f_i$  enforce the Newton's methods assumption of approximating the gradient through the Taylor's series expansion up to second order.

Once the Hessian is updated, a new iteration is started on Step 1. Since  $\mathbf{A}_i^{-1}$  is required to determine the optimal displacement direction, Eq. (3.15) can be reformulated in terms of  $\mathbf{B}_i$  as

$$\mathbf{B}_{i+1} = \mathbf{B}_i + \frac{\mathbf{d}_i\mathbf{d}_i^T}{\mathbf{d}_i^T\mathbf{g}_i} \left( 1 + \frac{\mathbf{g}_i^T\mathbf{B}_i\mathbf{g}_i}{\mathbf{d}_i^T\mathbf{g}_i} \right) - \frac{\mathbf{B}_i\mathbf{g}_i\mathbf{d}_i^T}{\mathbf{d}_i^T\mathbf{g}_i} - \frac{\mathbf{d}_i\mathbf{g}_i^T\mathbf{B}_i}{\mathbf{d}_i^T\mathbf{g}_i} \quad (3.16)$$

## 3.2 Evolutionary Optimisation Techniques

Frequently, in many real-world optimisation problems we face situations where the objective function and/or the constraints are not analytically tractable or lack a close form representation, e.g. if measuring an agent's performance in a game [113]. A common way to address such cases is to develop a close form approximation to the original problem that can be solved by classical MP methods. Nevertheless, there is a potential risk of oversimplifying the original problem, thus we could end up with a completely different problem. Evolutionary algorithms (EAs) are stochastic search techniques inspired by the process of evolution in living organisms [114–117]. They start with a initial set of potential solutions, known as population, randomly distributed in the solution space. Each potential solution in the population, called individual, is associated with a fitness value which measure its quality in solving the current problem. The population iteratively evolves into better solutions, close to the optimal, with the help of genetic operators. At each iteration, an offspring population is generated by recombination and/or mutation of the characteristics of the best individuals in the current population. Replacement of genetic material is performed over copies of the selected parents to avoid any disruption in their phenotype, in case they are selected again for mating. Then the offsprings fitness value is computed and only the best ones are allowed into the next generation. Sometimes the best performing parents are allowed to compete against the offsprings for selection into the next generation [118].

One of the key aspects in evolutionary algorithms is how to represent a solution that optimally fits the problem at hand. Generally, each individual is associated with a dual representation consisting of: (1) its genotype, interpreted as the container structure of its genetic material, and (2) its phenotype, understood as the actual expression of the characteristics as dictated by its genetic material. Usually a specific chromosome is designed to encode the characteristics of each problem, the population genotype, along with matching genetic operands. Often the mapping between an individual's phenotype and the numeric variables used by the objective function is not straightforward and an encoding function needs to be implemented. This mapping may introduce undesirable effects, such as unexpected nonlinearities, thus a representation design as close as possible to the numerical representation is strongly suggested [119]. Another relevant design component in EAs is the fitness function, which is responsible of assessing the performance of each individual. Frequently, the fitness function is closely related to, if not the same as, the objective function to be optimised. Variants to the aforesaid components give rise to multiple techniques, nevertheless, genetic algorithms, particle swarm optimisation and genetic programming are popular EAs among ML practitioners and will be briefly overview in the next sections.

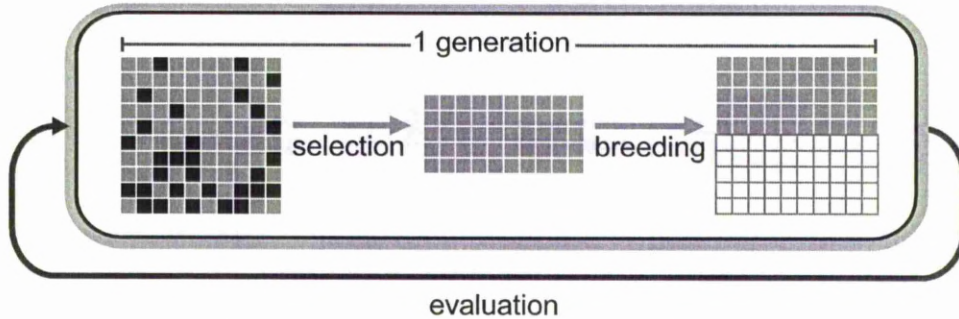


Figure 3.1: Main genetic algorithm loop, where a strong elitism policy is adopted by retaining half of the population.

### 3.2.1 Genetic Algorithms

With a sound theoretical basis developed by Holland [120], genetic algorithms (GA) are possible the most popular EA so far. They can be seen as stochastic directed search algorithms which use a binary alphabet to directly encode optimisation variables into a linear chromosome. The popularity of such representation is derived from the analysis proposed by the schema theorem. It states that GAs are near-optimal sampling strategies which increase the number of well-performing, short and low-order schemata (i.e. similarity templates that represent a subset of strings) as the population evolves. Nevertheless, a binary chromosome suffers an exponential growth with the increase in design variables. In this case, non-binary alphabets can be used with a slightly complexity increase in the genetic operations. A graphic representation of the main GA routine is depicted in Fig. 3.1

Mutation and crossover are the two genetic operations used by GAs to generate offsprings at each iteration of the algorithm. Crossover enables exploitation of the solution space by passing onto the next generation the characteristics of the best performing solutions. On the other hand, mutation makes possible to explore new solutions not contemplated in the current population by introducing controlled perturbations into the genetic material of selected individuals. Usually, a balance between mutation and crossover is desirable to efficiently explore the solution space. Such balance is controlled by means of a crossover probability  $P_c$ , that states the number of times crossover is selected over mutation whenever an offspring needs to be generated. Although, in the majority of implementations  $P_c$  is fixed, adaptive mutation rates can be used to promote high exploration at the start of a GA by setting a low  $P_c$  value, and gradually increasing it with the number of generations so that exploitation of good solutions is encouraged [121].

Several implementations of genetic operators have been developed to suit specific applications and/or problems [122]. A popular crossover implementation is  $\chi$ -point crossover for discrete representations, which splits each parental chromosome at

$\chi \in \{2k; \forall k \in \mathbb{Z}\}$  randomly selected points, and the genetic material is swapped correspondingly to generate two children. This type of crossover permits a trade-off control between distributional bias (i.e. production of offsprings with phenotype completely different from their parents by selecting a high  $\chi$  value), and positional bias (i.e. interchange of only distant alleles in a chromosome as a consequence of low  $\chi$  values). For real-valued representations, a widespread crossover known as *intermediate arithmetical crossover* constructs each allele in the offsprings following the formula

$$\begin{aligned}\mathcal{P}_{t+1}(i, k) &= r_k \mathcal{P}_t(i, k) + (1 - r_k) \mathcal{P}_t(j, k), \\ \mathcal{P}_{t+1}(j, k) &= (1 - r_k) \mathcal{P}_t(i, k) + r_k \mathcal{P}_t(j, k),\end{aligned}\tag{3.17}$$

where  $r_k \in [0, 1]$  is a random uniform number,  $\mathcal{P}_t(i, k)$  is the  $k^{th}$  allele of the  $i^{th}$  member of the population at generation  $t$ .

Every time mutation takes place, an allele is considered for replacement with probability  $P_m = 1 - P_c$ . For discrete representations, mutation replaces each allele with a random element from the given alphabet. While for real-valued representations, three classic mutation schemes are reported in the literature. *Uniform mutation* substitutes the  $k^{th}$  allele with a uniformly distributed random value between the interval  $[L_k, U_k]$ . *Gaussian mutation* adds a normally distributed random number  $N(0, r_k P_m)$  with zero mean and variance proportional to the mutation rate to each genome in the parental chromosome. *Adaptive non-uniform mutation* is based on a Bernoulli trial  $r$ , with  $p_s = 0.5$  probability of success. It decides to replace the  $k^{th}$  allele following the rule

$$\mathcal{P}_{t+1}(i, k) = \begin{cases} \mathcal{P}_t(i, k) + \Omega(t, U_k - \mathcal{P}_t(i, k)) & \text{if } r < p_s, \\ \mathcal{P}_t(i, k) - \Omega(t, \mathcal{P}_t(i, k) - L_k) & \text{otherwise} \end{cases}\tag{3.18}$$

where  $\Omega(t, a) = a \left(1 - r_k^{(1-t/N_{gen})}\right)$  is a mapping delivering a value in the range  $[0, a]$  at generation  $t$ , closing towards zero as we reach the final generation  $N_{gen}$ .

Being a stochastic search, convergence can not be guaranteed in a fixed number of steps, therefore a topping criterion needs to be defined. As with the majority of the EAs, three possible heuristics can be implemented as stopping criterion: (1) Stop when the maximum number of generations  $N_{gen}$  has been reached. (2) Run the algorithm until there is no significant difference between the best fitness value  $F_{best}$  and the average population fitness  $F_{avg}$ , i.e.  $|F_{best} - F_{avg}| \leq \xi$ . (3) Stop if the average cumulative change in fitness function values over  $N_{stall}$  generations is less or equal than a given tolerance  $\frac{1}{N_{stall}} \sum_{k=i-N_{stall}}^{i-1} |F_k - F_{k+1}| \leq \xi$ .

In summary, a GA can be described by the following iterative sequence

1. Start with a random initial population  $\mathcal{P}_t|_{t=0}$  with  $N_{ind}$  number of individuals.
2. Evaluate the performance of each individual by computing its fitness function  $f(\mathcal{P}_t(i))$ .

3. Rank each individual according to its fitness value and applied a selection scheme to get a set of stallions  $\mathcal{S} \subset \mathcal{P}_t$  for breeding.
4. For  $k$  from 1 to  $M = \varrho N_{ind}$ , build a new individual based on a Bernoulli trial  $r$  with  $P_c$  probability of success, following the rule
  - if**  $r \leq P_c$  **then**
    - generate  $\mathcal{P}_{t+1}(k)$  and  $\mathcal{P}_{t+1}(k + 1)$  from `CrossOver`( $\mathcal{S}(k), \mathcal{S}(k + 1)$ );
    - $k = k + 2$ ;
  - else**
    - generate  $\mathcal{P}_{t+1}(k)$  from `Mutate`( $\mathcal{S}(k)$ );
    - $k = k + 1$ ;
  - end if**

where `CrossOver` and `Mutate` are the corresponding crossover and mutation operators, and  $\varrho < 1$  is a scalar known as the generational gap ratio, which indicates the amount of elite individuals that will be passed unaltered to the next generation.

5. For  $k$  from  $M + 1$  to  $N_{ind}$ , insert  $\mathcal{P}_{t+1}(k) = \mathcal{S}_{best}(k)$ , where  $\mathcal{S}_{best}(k)$  is the set of  $(1 - \varrho)N_{ind}$  fittest individuals.
6. Set  $t = t + 1$ , and evaluate the performance of each individual in population  $\mathcal{P}_t$  by computing its fitness function  $f(\mathcal{P}_t(i))$ .
7. If the selected stopping conditions are met, return  $\mathcal{S}_{best}(1)$  as the approximate solution to the optimisation problem encoded in the fitness function, otherwise go to step 3.

### 3.2.2 Genetic Programming

Initially developed by Koza [123] with the propose of evolving computer programs, genetic programming (GP) is a strong and popular variation of GAs which employs parse trees as genotype structures. The flexibility of such representation formalism makes possible to evolve different knowledge abstractions, such as mathematical expressions, rule-based systems, finite-state machines, or regular expressions. Additionally to the genetic container, two more components need to be specified to completely define a population's genotype: a *terminal set* consisting of all the symbols leaf nodes will be allowed to take; a *function set*, which includes nonterminal symbols associated with functions and operators, that will interact with the input symbols. Two important conditions are imposed on the design of function and terminal sets, namely sufficiency and closure. Sufficiency states that together, function and terminal sets, must be able to generate a solution to the specific problem. Closure, guarantees a free interaction

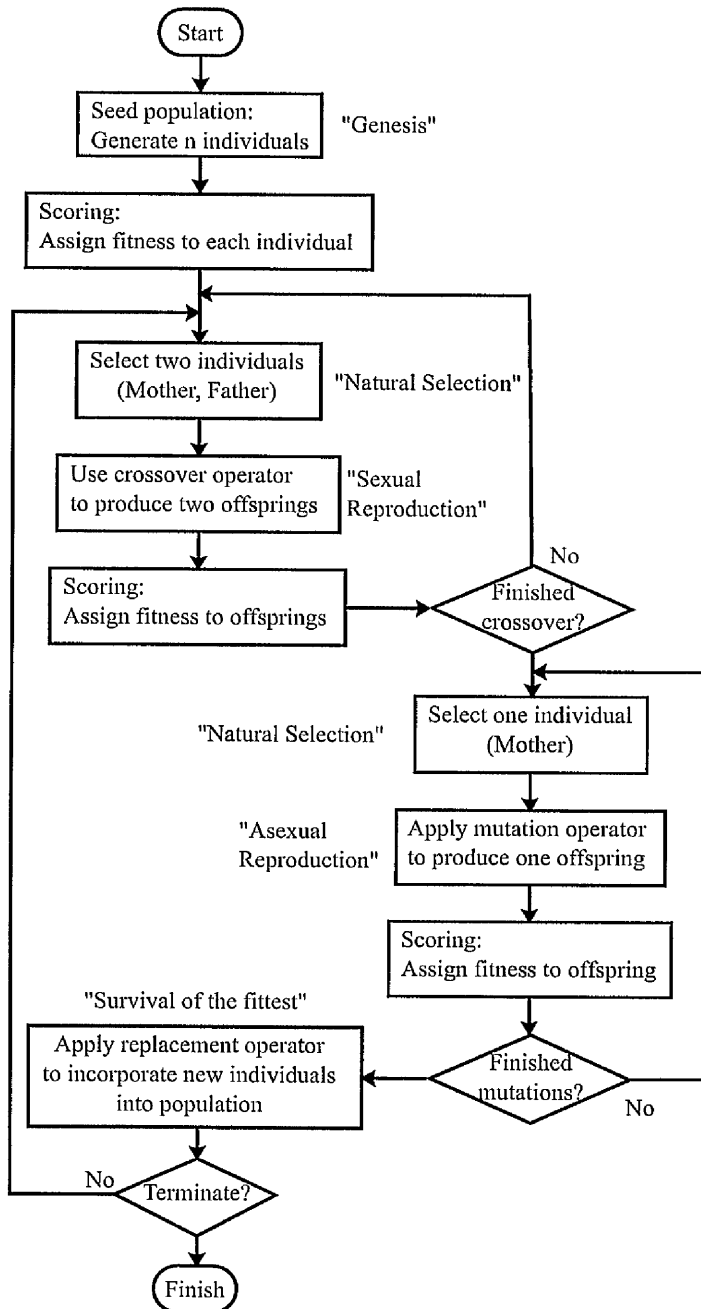


Figure 3.2: Flowchart representation of a GA with crossover and mutation operators.

between leafs and internal nodes by requiring that any member in the function set must be able to accept as input arguments any terminal, or output value from any function composition of other members in the function set. The number of input arguments a given member in the function set accepts is called *arity*, and it is reflected in the number of branches the corresponding internal node can grow.

The complex representation adopted by GP demands specific changes to the building process of the initial population, and also to the mutation and crossover behaviour. Given an initial maximum depth  $d_{max}$ , an initial population can be constructed following one of three available methods [124]

**Full method** generates balanced trees with branches with equal depth. Starting from the root node  $d = 1$ , it adds a randomly selected function node until it reaches  $d = d_{max} - 1$ . At depth  $d = d_{max}$ , a terminal node is randomly selected. The nodes are added in the same order they would be visited by a depth-first search.

**Grow method** randomly selects each new node from the union of the function and terminal sets, except at depth  $d = d_{max}$  where only terminal nodes are allowed. This method generates highly unbalance trees with variable depth between  $[1, d_{max}]$ .

**Ramped half-and-half** builds half of the total number of individuals  $N_{ind}$  using the grow method, and the other half using the full method. For each half, the depth of each individual is linearly increased, with the first individual having depth  $d = 2$ , and the last one with depth  $d = d_{max}$ . The result is a diverse population with trees of different shapes and depths.

Once an initial population has been created, each individual is decoded and evaluated according to the fitness function. Following, a population subset for mating is defined with the help of selection mechanism used in others EAs. Nevertheless, offsprings generation in GP differs significantly from others EAs. Typically, crossover is performed via *subtree crossover* as shown in Fig. 3.3a. Given two parents, it randomly selects a node as crossover point in each parent, and then the corresponding subtrees at those points are swapped. Figure 3.3b shows an example of *subtree mutation* operator, where a subtree rooted at a randomly selected mutation point is replaced with a newly generated tree. This mutation operator is also known as "*headless chicken*" crossover, because it can be implemented as a subtree crossover between the selected individual and a random generated tree.

An interesting side effect of unrestricted subtree mutation and crossover is their ability to produce extremely large individuals without a considerable increase in their fitness value. This phenomenon is known as *bloat* and several strategies have been proposed [125] to alleviate this problem. A wide spread technique restricts the size or depth of generated offsprings to an upper limit [123]. Once an offspring is generated, its size or depth is measured, if the offspring is within the set limits, it is pass onto the



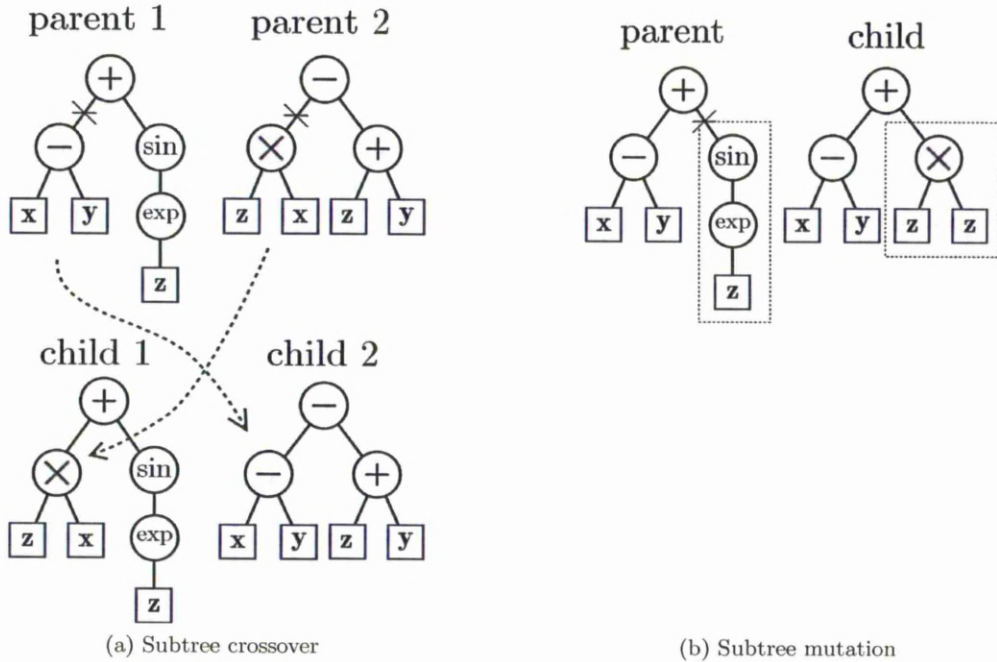


Figure 3.3: Basic subtree genetic operators for genetic programming using the terminal set  $\{x, y, z\}$ , and the function set  $\{+, -, \times, \sin, \exp\}$  with corresponding arities  $[2, 2, 2, 1, 1]$ . Crossover and mutation points are indicated by a 'x' on the selected branch.

next generation, otherwise one of its parents is returned. Unfortunately this technique tends to favour parents prone to violate the imposed restrictions, by copying them unaltered into the next generation. Two alternatives exist to mitigate the adverse effects of size and depth limits. The first option is to let the evolution process eliminate an overgrown offspring by assigning it a low fitness value, close to  $-\infty$  if we are dealing with a maximisation problem. The second one is to try the corresponding genetic operator again with the same parents, but different mutation or crossover points, or with new selected parents. Other option for bloat control is the use of dynamic depth or size limits [126], where every time an overgrown offspring is generated, if its fitness value is better than the best solution found so far, the dynamic limit is set to the size or depth of the offspring.

### 3.2.3 Particle Swarm Optimisation

Particle swarm optimisation (PSO) represents a different paradigm to the Darwinian survival of the fittest advocated by GA and GP. It is inspired by the social interactions present in groups of animals, moving or migrating in some direction. The algorithm assumes a set of  $m$ -dimensional points, also known as the swarm, which explore the solution space by iteratively adjusting the position of each point, referred as a particle, towards the position of its own best and the best particle in the swarm. Besides its

position  $\mathbf{x}_i$ , each particle is associated with a velocity  $\mathbf{v}_i$  which indicates how far the particle is allowed to travel. Given a maximum number of particles  $N$ , and assuming the quality of a solution encoded in particle  $\mathbf{x}_i$  can be directly assessed by  $f(\mathbf{x}_i)$ , where  $f$  is the function to be optimized, PSO starts by creating randomly distributed particles with aleatory velocities. The personal best position of the  $i^{\text{th}}$  particle is kept in variable  $\mathbf{p}_i$ , whereas the global best position is stored in  $\mathbf{p}_g$ . The iterative process in PSO can be described as follows

For each particle  $\mathbf{x}_i$ ,  $i = 1, \dots, N$  in the swarm:

1. Evaluate the objective function  $f(\mathbf{x}_i)$
2. If  $f(\mathbf{x}_i) < f(\mathbf{p}_i)$ , then  $\mathbf{p}_i = \mathbf{x}_i$
3. Get the global best as  $\mathbf{p}_g = \operatorname{argmin}_{i=1, \dots, N} f(\mathbf{p}_i)$
4. Update every component of the velocity vector  $\mathbf{v}_i = [v_{i1}, \dots, v_{im}]$  as follows

$$v_{ij} = wv_{ij} + c_1 z (p_{ij} - x_{ij}) + c_2 z (p_{gj} - x_{ij}), j = 1, \dots, m \quad (3.19)$$

where  $c_1$  and  $c_2$  are constant coefficients controlling how far the particle will travel every time its position is updated,  $z \in [0, 1]$  is a random positive number, drawn from a uniform distribution, and  $w \in [0, 1]$  is an inertia weight.

5. Update the position vector  $\mathbf{x}_i = [x_{i1}, \dots, x_{im}]$  employing

$$x_{ij} = x_{ij} + v_{ij}, j = 1, \dots, m \quad (3.20)$$

Once every particle has been updated, if a given termination criterion is met, return the global best, otherwise repeat the above process.

An additional variation to the standard PSO algorithm is given by time varying inertia weights (TVIW-PSO) where the inertia weight  $w$  in Eq. 3.19 is replaced by a time dependant version described as

$$w = (w_0 - w_T) \frac{T - t}{T} + w_T \quad (3.21)$$

where  $w_0$  is the initial value of the inertia weight,  $w_T$  is a desired upper-bound to be reach at the final generation  $T$ . Thus for each iteration  $t$ , the value of the inertia weight is linearly decreased.

### 3.3 Evolutionary Optimisation in Pattern Recognition

Over more than a decade, evolutionary algorithms (EAs) have been studied and applied as powerful design, search and optimisation techniques. Despite their differences, all evolutionary algorithms are heuristic population-based search procedures that incorporate random variation and selection. They have been applied to diverse areas such as biology [127–129], medicine [130–132], finance [133–135], law [136–138], engineering [139–141], software engineering [142–144], oceanography [145], communications [146–148], and pharmacology [149–151], among others. In the field of machine learning, EAs have been employed as designing tool, to perform system identification for the tasks of classification, regression, feature extraction, feature selection, and clustering. An extensive amount of research has been devoted to the previously mentioned tasks, and has been thoroughly summarised in corresponding surveys [152–155]. In most of the works, EAs employ a grey-box approach to system identification, where a structure is assumed for a generic model, given by the targeted pattern recognition algorithm. The free parameters of the model are encoded and estimated by the selected EA. The term *genetic-based machine learning* (GBML) algorithms was coined in [156] referring to applications of EAs to machine learning tasks, and will be adopted for the remainder of this section.

For the problem of classification, three types of representations are regularly used by GBMLs: decision trees [7], classification rules [157], and discriminant functions [1]. They often are encoded into tree-based chromosomes related to GP, or fixed-length string chromosomes (also known as linear chromosomes) required by GAs. Although linear chromosomes have been used to encode all the aforesaid representations [158–160], they have proven to be a more suitable formalism for classification rules. While tree-based chromosomes are more popular among evolution of decision trees and discriminant functions. The variable nature of each individual under the tree-based representation implicitly defines a feature selection process, where only some of the original features will be present in each individual. For instance, the number of features to be tested will depend on the number of internal nodes when evolving axis-parallel decision trees (e.g. [155,161–163]), while the hyperplane at each node in an oblique decision tree will be a function of only a few input features (e.g. [164–167]).

Independently of the encoding scheme selected, GBMLs can be classified according to two types of architectures [169]. *Pittsburgh architecture* encodes a full solution (classifier) into a single chromosome, evolving a population of potential solutions which delivers the best individual in the final population as output of the algorithm (Fig. 3.4 right). In *Michigan architecture*, the solution is encoded into the whole population, where each individual, besides competing for reproduction, also complements and cooperates with the rest of the population (Fig. 3.4 left). When a predefined number of generations elapses or when some other termination criterion is met, the final popula-

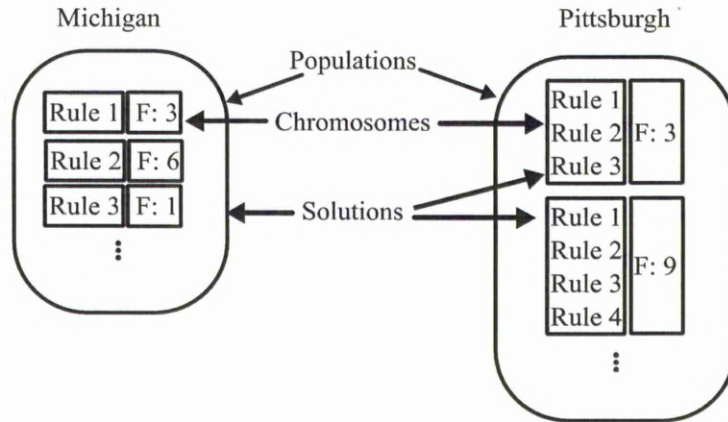


Figure 3.4: Michigan and Pittsburgh architectures for rule-based systems. F:x indicates the fitness for each individual. (Adapted from [168])

tion is decoded to build a single classifier which is delivered as output of the GBLM. The design of a fitness function is more complex for Michigan than for Pittsburgh architectures, since it must measure the degree to which one individual contributes to the solution, but as well how it interacts with the rest of the population. On the other hand, Pittsburgh architectures often need more elaborate genetic operators and chromosome structures. Besides solution encoding, both architectures differ in the style of learning adopted. While Pittsburgh systems operate in batch mode, processing at once all the available data, Michigan systems perform incremental learning, updating the population every time a new example is presented. Thus, the former strategy is commonly used for offline learning, whereas the latter is more popular for online learning.

Leaving aside the previously described architectures, [154] proposed three additional categories to classify GBMLs for rule induction. *Iterative rule learning* (IRL) algorithms build an ordered list of rules. The  $i$ th rule is evolved using the training examples not covered by the previous rules. After evolution, the examples covered by the current rule are removed and the EA is called again to generate a new rule. This process stops when there is no more training examples. Representative algorithms in this category comprise supervised inductive algorithm [170], and hierarchical decision rules [171]. *Genetic cooperative competitive learning* (GCCL) algorithms differ from Michigan style systems in that they use a generational GA to evolve a set of rules encoded into the population, but retain the incremental learning and solution encoding. Also, they make use variable length individuals, avoiding "don't care" conditions, and allow for nominal feature representation. GCCL examples include co-evolutionary rule extractor [172], organizational co-evolutionary algorithm for classification [173], and coverage-based genetic induction [174]. *Hybrid evolutionary decision trees* (HEDT) are hybrid approaches that represent a rule set as a decision tree, which is evolved by means of generational GAs, using linear chromosomes. They inherit the batch mode

learning from Pittsburgh architectures, and the use of oblique decision rules from its tree representation. Although in general XCS exhibits high generalisation performance when comparing to other algorithms because its solution encoding allows a compound interaction between diverse rules, it is prone to deliver less interpretable solutions.

Other than decision trees and discriminant functions, GBMLs have been used to evolve more complex classifiers. In [175] an hybrid classifier, consisting of a classification rule set and a discriminant function, is evolved using a two-stages GBML algorithm. First a classification rule set is evolved using a Michigan architecture in the first phase, which reduces the training set to examples not covered by the current population. Second, a single-threshold discriminant function is evolved using the reduced training set. In this way, when a test instance is not covered by the evolved rule set, the discriminant function is set a default rule. Evolutionary approaches have also been applied to the induction of support vector machines (SVM) [176–180], where each individual in the population encodes a kernel function, and their fitness is measured by the classification performance of the SVM each one builds. Another classifier representation for GBMLs is the k-nearest neighbour classifier. In this case, EAs are designed to optimised specific components of the classifier, such as weighting functions to scale class counts [181], distance functions to compute the neighbours of a sample [182], or the prototypes in their training set [183]. Other examples of evolved classifiers are kernel nearest neighbour [184], and variable predictive models [185]. A more sophisticated Michigan architecture, employing other classifiers representation, has been used by GBMLs evolving ensemble classifiers [19,186–188]. An ensemble classifier combines a set of weak models in order to produce a strong model, thus each individual in the population encodes a full classifier which cooperates with the rest of the population to build the ensemble.

## Evolutionary Optimisation for Feature Processing

As previously mentioned, the induction of classifiers by means of evolutionary approaches implicitly defines a feature selection process. Additionally, when the selected chromosome encoding combines input features through arithmetic operators, an underlying feature construction process is suggested. Following this simple ideas, induction of preprocessing methods by means of evolutionary algorithms can be defined as a special category of GBMLs where a subset of the original features is selected/constructed to boost a targeted classification algorithm. A vast amount of work has been dedicated to this task [153], and can be grouped into two categories, namely wrapper and filter systems. Wrapper systems make use of the subsequent classifier algorithm to evaluate potential solutions within the evolution process. While filter systems employ some other statistical information criterion to measure the quality of the subset, such as mutual information, cosine norm, Pearson's, Spearman's or Kendall's correlation [42]. Usually

the number of selected/constructed features is predetermined by the user, which are employed alone [32, 189, 190] or in combination with the original feature set [191] to train a given classifier.

The preferred encoding scheme for feature extraction in GBMLs is the tree-based chromosome because its flexibility and straight function evaluation. However, a few works using linear chromosomes have been proposed [26, 54, 192, 193]. For example, [54] encodes into a linear chromosome the angles of an affine transformation that projects input features into a subspace of low dimensionality. Often, in GBMLs for feature extraction, the fitness function is implemented with the help of ROC-related concepts such as sensitivity, accuracy, precision, hit rate, or a linear combination of them, estimated via cross validation. On the other hand, various fitness functions have been proposed for filter approaches, such as information gain, the gini index, the chi-square index [191], between-class scatter [32, 189], and information entropy [190]. In order to evolve a determined number of features, the selected GBML is run several times or a multiple gene chromosome is adopted, where each gene represents a different feature. A number of different classifiers have been applied as the core of wrapper approaches such as k-NN, generalized linear machine [194], maximum likelihood, C4.5, and naive Bayes.

In contrast, GBMLs for feature selection favour the use of fixed-length string chromosomes. The common genotype-phenotype mapping for a binary alphabet consists on selecting the  $i$ th feature whenever the corresponding bit is on (e.g. [17]). More complex mappings exist for integer alphabets, for instance in [186] the chromosome is divided into two parts, the first part corresponds to the feature encoding where each character is in base-8, interpreted as three binary flags indicating the classifiers using the corresponding feature (e.g. the character 5 in the third position indicates classifiers 1 and 3 use the third input feature); the last three characters denote the type of classifier, selected from linear discriminant, quadratic discriminant, or logistic regression classifier. Even though linear chromosomes are easier to implement and provide a straightforward genotype-phenotype mapping, for high-dimensional data they may require large amount of memory. Tree-based chromosomes provide a more compact structure and constitute a good alternative for such cases [195]. However, alternative representations such as axis-parallel decision trees and univariate rule encodings need to be adopted, forcing the genotype-phenotype to ignore any node different from the original set of attributes. Although GBMLs for feature selection are time-consuming compared to non-evolutionary ML methods, a better classification performance has been reported in most of their applications [196–199].

## Chapter 4

# Automated Induction of Projection Pursuit Indices

### 4.1 Introduction

In general, it is known that the performance of an entire pattern recognition system becomes affected by the interactions between the feature extraction and the classification stages [1]. Thus, the overall design becomes a complex model selection problem where a suitable feature extraction/classification pair needs to be found. The complexity of this optimisation problem in [19] was tackled with a GA used to build an optimal ensemble classifier. Other relevant work on feature extraction/classification pair selection was described in [186] where an ensemble of three classifiers was used in conjunction with a subset of features selected by a GA. A robust technique to synthesize a complete recognition system was also proposed in [187] where GP built the structure of several feature detectors and then a GA was applied to select a subset of them, followed by a perceptron classifier performing multiclass recognition. Cooperative evolution of artificial neural networks ensembles was used to improve the generalization performance of classification systems and avoid long training times overfitting in [45].

This chapter introduces a GP-based framework for automatic induction of projection pursuit indices. The referred framework optimises a set of hyperparameters for a given classifier, and simultaneously obtains an optimal feature extraction/classifier pair for a given classification problem. A complex set of function and terminal nodes was designed based on robust high-order statistics and shape-shifting approximations to divergence and entropy, in order to grant the GP search high expressive power and generate new PP indices not previously considered in the field. Its search space is broad enough to include the most popular indices among the existing literature. Although the proposed algorithm has the ability to combine existing indices, it is not restricted to a simple linear combination of them, but is capable of building any complex and arbitrary but meaningful function composition of the members from the function set. The targeted index function is used in PP to get a transform matrix that serves the

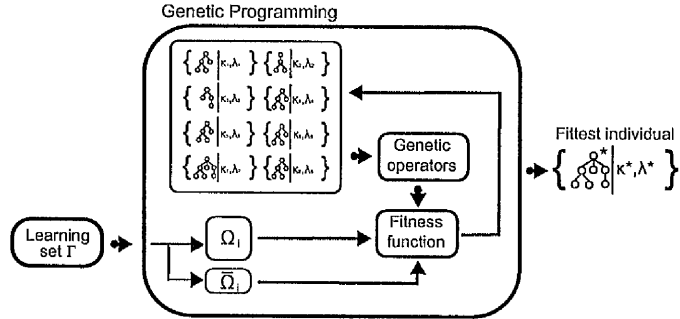


Figure 4.1: Evolutionary model selection. The learning set  $\mathcal{T}$  is fed into the GP module to get an optimal model  $\{\mathcal{S}^*, \theta^*\}$  that suits the problem at hand. This module realises the model selection procedure where the search is guided by the fitness function.

feature extraction stage. Because the projection index  $\mathcal{S}$  delivers a linear transform, we can think of it as a way of modelling or compacting the type of information the features carry. The number of dimensions of the extracted feature space is automatically estimated by a Bayesian criterion.

Once dimensionality reduction has been performed, classification is applied in the reduced subspace by employing a k-nearest neighbours (kNN) classifier based on a Minkowski distance. Therefore a two-level optimisation problem needs to be solved. The first level consists on the selection of an optimal projection index  $\mathcal{S}$  and optimal hyperparameters for the selected classifier by means of GP. The second level corresponds to the optimisation of the basis vectors of the transform  $\mathbf{P}^*$  in Eq. (2.34), by a simple gradient descent algorithm.

## 4.2 Proposed Evolutionary Learning System

Ideally, in order to efficiently design an projection index that best suits the application at hand, a guided search would be performed over all possible function compositions of a basic set of functional primitives. In the current literature [7, 10, 200], the search for an optimal feature extraction/classifier pair is often performed over a grid defined by all possible combinations of the available methods for each application, driven by an estimation of misclassification error. In addition to the complexity of the grid search, one needs to consider the hyperparameters related to the classifier itself (such as number of hidden nodes for multilayer perceptrons, covariance mixing parameters for discriminant analysis, or regularizers for support vector machines). It is desirable to fine-tune those parameters to suit the selected feature extraction method, and hence they have to be included as part of the model [201], but this adds further complexity to the search space. In order to avoid the computational burden involved in grid search, evolutionary methodologies have been successfully applied to address automatic design of classification systems [44, 46, 186, 187, 202]. GP, for instance, has proven to be a



suitable tool for feature generation [32,203], evolution of polynomial feedforward neural networks [204], and feature selection [205].

This work proposes a novel framework based on GP which performs model selection driven by an estimation of out-of-sample generalization error. In the following the proposed evolutionary framework will be labelled as evolutionary projection pursuit (EPP). The proposed framework follows a Pittsburg architecture to facilitate the design of the fitness function. The population in the GP algorithm consists of potential models that through evolution will be modified to increase their classification accuracy. Each model is encoded as a pair  $\{\mathfrak{S}, \theta\}$ , where  $\mathfrak{S}$  is a potential index function represented as a tree structure, and  $\theta$  is a set of hyperparameters for a given classifier. Fig. 4.1 shows a graphic representation of the proposed model selection algorithm.

A simple kNN classifier was selected as part of the evolved system in order to highlight the feature selection part, since a more elaborate classifier may not benefit from the complex feature extraction stage considered in this work. Nevertheless, EPP is not restricted to the use of a specific classifier, and a more powerful one could be used with the corresponding modifications in the fitness function. The referred classifier is instantiated for each potential model as the number of nearest neighbours  $k$  and the coefficient for the Minkowski metric  $\lambda$  used to locate the neighbours are evolved concurrently with the projection index, therefore  $\theta = \{k, \lambda\}$ .

#### 4.2.1 Fitness Function

Since the evolutionary system selects the best model, a suitable way for comparing the different potential models in the GP population needs to be provided. The chosen model selection criterion was the out-of-sample estimation error via cross-validation (CV); thus the evolution process can be modelled as follows:

$$\begin{aligned}
\min \quad & F(\mathfrak{S}, \theta; \Upsilon) = \sum_{i=1}^h \sum_{(\mathbf{x}_j, y_j) \in \bar{\Omega}_i} L(y_j, \varphi_{\Omega_i}(\mathbf{x}_j; \mathbf{P}^*, \theta)) \\
\text{s.t.} \quad & \mathbf{P}^* = \underset{\substack{\mathbf{P} \in \mathbb{R}^{d \times m}, \\ \mathbf{P}^T \mathbf{P} = \mathbf{I}}}{\text{argmin}} \left\{ \mathfrak{S}(\mathbf{x}^T \mathbf{P})_{\mathbf{x} \in \Omega_i} \right\} \\
& \lambda > 0 \\
& k \in \mathbb{N}^+
\end{aligned} \tag{4.1}$$

where  $\Upsilon = \{\Omega_i \cup \bar{\Omega}_i\}$  is a given learning set,  $L(\cdot)$  is the 0-1 loss function between the label  $y_i$  and the prediction made by the classifier  $\varphi_{\Omega_i}(\mathbf{x}_j; \mathbf{P}^*, \theta)$  using sample  $\mathbf{x}_j \in \bar{\Omega}_i$ , parameters  $\theta = \{k, \lambda\}$ , and trained with prototypes in  $\Omega_i$ . The imposed constraints in Eq. (4.1) were designed to restrict the Minkowski distance coefficient to the positive reals, and the number of neighbours  $k$  to the positive integers  $\mathbb{N}^+$ . The projection index  $\mathfrak{S}$  and the optimal projection matrix  $\mathbf{P}^*$  were placed as inputs to the classification stage,

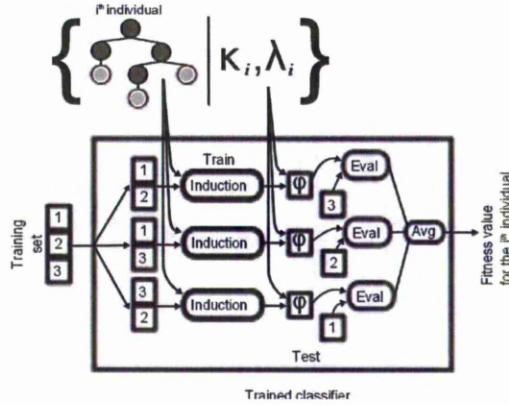


Figure 4.2: GP fitness function involving 3-CV and induction of the feature extraction stage.

since previous to the prediction process, the sample  $\mathbf{x}_j \in \bar{\Omega}_i$  and the training prototypes in  $\Omega_i$  must be projected onto the subspace spanned by  $\mathbf{P}^*$ .

In order to obtain the out-of-sample estimation error for each individual, first the given points  $\Upsilon$  are divided into training  $\Omega_i$  and testing set  $\bar{\Omega}_i$ . The training set is used to find the optimal projection matrix  $\mathbf{P}^*$ , which in turn is used to project both  $\Omega_i$  and  $\bar{\Omega}_i$ . Then,  $\bar{\Omega}_i$  is used to evaluate the performance of the potential feature extraction/classification pair, using  $\Omega_i$  as the prototypes of kNN and the  $k$  and  $\lambda$  as the kNN parameters. This process is repeated over each fold of a  $h$ -CV (a fix  $h = 3$  was set for all experiments) in Eq. (4.1), and the performance over the testing sets averaged to form the estimated error. The reason behind the use of  $h$ -CV as a way to estimate the out-of-sample generalization error is to avoid overfitting [7].

Fig. 4.2 gives a system overview of Eq. (4.1). The block named “Induction” performs PP using as projection index the function associated with the feature extraction part of the  $i^{th}$  individual (i.e.,  $\mathfrak{S}$  function). When using a non-prototype based classifier, it is at this level where the training of the classifier will take place, as the optimisation of  $k$  and  $\lambda$  in this case, which are the hyperparameters related with the classification part in each  $i^{th}$  individual. Subsequently, the out-of-sample classification error is estimated via CV and returned as a fitness measure to the GP module, which selects the optimal tree and classifier parameters using this fitness measure.

#### 4.2.2 Index Optimisation

Equation (4.1) can be interpreted as a bilevel optimisation problem [206] which is solved by two optimisers. The first level minimisation problem is accomplished with a GP algorithm, which searches for the optimal PP index and kNN parameters given a training set. The second level minimisation problem is carried out with a gradient algorithm, which finds the optimal basis vectors  $\mathbf{P}^*$  given the corresponding PP index

$\mathfrak{S}$  as in Eq. (2.34). Several optimisation methods have been used in PP to optimise  $\mathfrak{S}$ , including hybrid GAs [67], simulating annealing [62] and PSO [26]. Nevertheless, hill-climbing algorithms remain as the most popular methods used in PP. In this thesis, the BFGS variant of the Quasi-Newton algorithm was selected to optimise the index function in each potential model because it converges relatively fast, it uses finite-differences to compute the derivative of the objective function, and avoids inversion of the Hessian. Additionally, BFGS was also used satisfactorily for probability density function (pdf)-based indices in [27].

It has to be noted that because gradient algorithms can be prone to local optima and are often sensitive to steep nonlinearities of the objective function, an individual index  $\mathfrak{S}$  may not be optimised effectively, producing projections  $\mathbf{P}^*$  that are suboptimal and do not illustrate the capacity of  $\mathfrak{S}$  realistically, yielding thus a weak fitness response. Because of this, the proposed two level optimisation design not only searches for an index able to minimise the classification error, but also an index easily optimizable by a gradient method. The latter can be seen as an implicit optimisation objective of the first level optimisation process.

### 4.2.3 Dimensionality Control

The number of columns in the matrix  $\mathbf{P}^*$  defines the dimensionality of the extracted features. When using PP as feature extraction, there are two ways to build matrix  $\mathbf{P}^*$ , by means of parallel projection pursuit (PPP) or using sequential projection pursuit (SPP). Parallel projection pursuit attempts to jointly optimise every component in the projection matrix. Although PPP was defined earlier than SPP [56], the expensive computation of multivariate integrals implicitly defined in the projection index restricted its use.

SPP finds the best 1-D projection  $\mathbf{p}_j^*$  (corresponding to the  $j^{th}$  column of  $\mathbf{P}^*$ ) measured by a given projection index, and then removes the contribution of such projection from the original feature space [67]. This removal procedure can be carried by projecting the original space onto the orthogonal complement of the projections found so far [50], and then the index is optimised again over the residual space to get the next column  $\mathbf{p}_{j+1}^*$  of  $\mathbf{P}^*$ . This process is iterated until a predefined number of factors are found or the rank of the new data matrix approaches zero. Given its well established use in the current literature, this thesis opts for the SPP style for constructing the projections one dimension at a time.

Regarding the orthogonality restriction imposed over the components of the projection matrix, the method proposed in [84] is adopted to implement an efficient deflation scheme. It is based on the assumption that once the  $i^{th}$  projection vector  $\mathbf{p}_i^{(j)} = [p_{i,1}, \dots, p_{i,j}]^T$  ( $j = d - i + 1$  and  $i = 1, \dots, m$ ) has been found then, there is at least one component  $p_{i,q} \neq 0$  such that any vector  $\mathbf{v}_l = [v_1, \dots, v_{j-1}]^T$  with  $v_l = 1$  ( $l \neq$

$q$ ),  $v_q = -p_{i,l}/p_{i,q}$ , and  $v_k = 0$  ( $k \neq q, l$ ), is orthogonal to  $\mathbf{p}_i^{(j)}$ . The set of vectors  $\{\mathbf{v}_l\}$  for  $l = 1, \dots, q-1, q+1, \dots, j$  form a basis defining the orthogonal space of  $\mathbf{p}_i^{(j)}$  as

$$\mathbf{Q}^{(j)} = \left[ \begin{array}{ccc|ccc} \mathbf{I}_{(q-1)} & & & \mathbf{0}_{(q-1) \times (j-q)} & & \\ \hline -\frac{p_{i,1}}{p_{i,q}} & \dots & -\frac{p_{i,q-1}}{p_{i,q}} & -\frac{p_{i,q+1}}{p_{i,q}} & \dots & -\frac{p_{i,j}}{p_{i,q}} \\ \hline \mathbf{0}_{(j-q) \times (q-1)} & & & \mathbf{I}_{j-q} & & \end{array} \right]. \quad (4.2)$$

Its orthonormal version is computed by means of the Gram-Schmidt process and denoted as  $\mathbf{Q}_\perp^{(j)}$ . Such transform is used to compute the reduced search space for  $\mathbf{p}_{i+1}^{(j)}$  as follows

$$\mathbf{Z}_{i+1} = \mathbf{X} \mathbf{Q}_\perp^{(d)} \mathbf{Q}_\perp^{(d-1)} \dots \mathbf{Q}_\perp^{(j)} = \mathbf{X} \prod_{k=d}^j \mathbf{Q}_\perp^k. \quad (4.3)$$

As can be seen, the search space  $\mathbf{Z}_{i+1}$  for the next projection vector is one dimension lower than  $\mathbf{Z}_i$ , therefore  $\mathbf{p}_{i+1}^{(j)}$  will be one component shorter than  $\mathbf{p}_i^{(j)}$  as indicated by  $j$ . To recover each  $\mathbf{p}_i^*$  from its deflated version  $\mathbf{p}_i^{(j)}$ , we have

$$\mathbf{p}_i^* = \prod_{k=d}^{j+1} \mathbf{Q}_\perp^k \mathbf{p}_i^{(j)} \quad i = 2, \dots, m. \quad (4.4)$$

The use of this method has the advantages of being faster than Gram-Schmidt based deflation scheme [84], reducing the computation and guaranteeing uncorrelatedness. A disadvantage is the need to store every  $\mathbf{Q}^{(j)}$ , so that the projection matrix  $\mathbf{P}^*$  can be built.

To automatically determine the number of projections in the SPP procedure, a recently proposed stopping criterion is employed, which relies on Bayesian model selection [21]. It is based on the fact that the remaining structure of the residual search space is decreased as the number of components increases. The stopping criterion is defined as

$$\mathbf{B} = \left( 2^{n \times [\mathfrak{S}(\mathbf{X} \cdot \mathbf{P}_{i-1}^*) - \mathfrak{S}(\mathbf{X} \cdot \mathbf{P}_i^*)]} + 1 \right)^{-1} \quad (4.5)$$

and includes the  $i^{\text{th}}$  projection component  $a_i^*$  if  $\mathbf{B}$  is bigger than a predefined threshold  $\delta$ , otherwise SPP stops with  $i-1$  projection vectors in  $\mathbf{P}^*$ .

### 4.3 Evolutionary Framework Language Definition

This section elaborates on the definition of a set of words and its formation rules to build finite strings that will be treated as potential projection indices. An extremely important characteristic of the proposed system is its expressive power, understood as the ability to generate most of the existing indices. This depends on the proposed function and terminal sets, since they will hold the building blocks available to the GP.

To make the present approach as general as possible, the GP module should have adequately high expressive power to be capable of discovering the definition of any of the aforementioned PP indices. Therefore, two very flexible function and terminal sets

Table 4.1: Language Definition.

(a) Function Set.

Functions	Arity	Description	Parameters
$+, -, *, /, \text{pow}$	2	Addition, subtraction, multiplication, division and exponentiation	
$m_s$	2	$s$ sample moment	$\mathbf{z}, s$
D	5	Rényi divergence	$\mathbf{z}, \rho, \tau, \nu, \xi$
H	2	Rényi generalized entropy	$\mathbf{z}, \rho$
$S_B$	2	Between-class scatter matrix	$\mathbf{Y}, \mathbf{z}$
$S_W$	2	Within-class scatter matrix	$\mathbf{Y}, \mathbf{z}$
$\mu_c$	3	Mean of the $c^{\text{th}}$ class	$\mathbf{Y}, \mathbf{z}, c$
$\sigma_c$	3	Variance of the $c^{\text{th}}$ class	$\mathbf{Y}, \mathbf{z}, c$
$Q_n^c$	4	$n^{\text{th}}$ quartile of the $c$ class	$\mathbf{Y}, \mathbf{z}, c, n$

(b) Terminal Set.

Terminals	Description
$\mathbf{z}$	Vector of projected data
$\mathbf{Y}$	Vector of class labels
$\eta$	Ephemeral random constant (e.g. $\rho, \tau, \nu, \xi, s, c, n$ )

are provided. First of all, basic arithmetic operands are needed in order to provide the GP with the basic tools to combine more complex functions. These are included in the function set, as displayed in the first row of Table 4.1a. To facilitate pdf-based indices, two functions are considered to construct valid approximations. The first one is Rényi's generalized divergence of order- $\rho$

$$D(\hat{f}, g; \rho) = \frac{1}{\rho - 1} \log \left( \sum_{i=1}^n \frac{\hat{f}_i^\rho}{g_i^{\rho-1}} \right) \quad \rho > 0 \quad (4.6)$$

where  $\hat{f}_i \equiv \hat{f}(\mathbf{z}_i)$  is the projected data probability estimate evaluated at  $\mathbf{z}_i$  and  $g_i \equiv g(\mathbf{x}_i)$  is the reference pdf evaluated at  $\mathbf{x}_i$ . This function measures divergence from a reference density while displaying interesting properties when  $\rho$  is varied [207]. It can potentially deliver the Kullback-Leibler divergence when  $\rho \rightarrow 1$ , thus allowing to approximate  $\mathfrak{S}_4$  in Table 2.1.

The flexibility embedded into Rényi divergence allows to define what it is considered as an *uninteresting* projection by means of the reference pdf. In [208] arguments were given for the use of Student-t distribution as another uninteresting pdf when robustness against outliers is required. Following a similar reasoning, divergence from a generalized extreme value (GEV) distribution is considered in this work, since its shape parameter

$\xi$  governs the tail behaviour of the distribution. This property is highly suitable for the proposed evolutionary system, as distribution specific parameters are included into the evolution process. In this way, it is possible to discover the ideal distribution density shape that will define what is considered as interesting or uninteresting projections.

Considering entropy as an efficient way to approximate divergence from gaussianity [60], the second function adopted to approximate pdf-based indices is Rényi order- $\rho$  entropy defined as

$$H(\hat{f}; \rho) = \frac{1}{1-\rho} \log \left( \sum_{i=1}^n \hat{f}_i^\rho \right); \rho \geq 0. \quad (4.7)$$

This function acts as a generalization of Shannon entropy. Due to its morphing characteristics conferred by the  $\rho$  coefficient, it can potentially deliver  $\mathfrak{S}_3$  as  $\rho \rightarrow 1$ , and allows us to approximate  $\mathfrak{S}_1$  when  $\rho = 2$  (see Table 2.1). As an example of pdf-based index construction, Fig. 4.3(e) and (d) shows the tree representation of  $\mathfrak{S}_1$  and the first term in  $\mathfrak{S}_4$  built with the proposed function and terminal sets.

To build moment-based indices, the  $s^{\text{th}}$  sample central moment defined as

$$m_s(\mathbf{z}) = E[(\mathbf{z} - E[\mathbf{z}])^s] \quad (4.8)$$

where  $E[\cdot]$  is the expectation operator, is included as another member of the function set. Together with the arithmetic operands and the terminal set, it allows the GP to come up with the definition of the indices in Table 2.2. In Fig. 4.3(b) and (c), the tree representation for indices  $\mathfrak{S}_6$  and  $\mathfrak{S}_5$  can be observed in terms of the proposed function and terminal sets so far.

As for the class-information-based indices, class means and variance, defined as  $\mu_c = \mu(c, \mathbf{Y}, \mathbf{z}) = E[\{\mathbf{z}_i : y_i = c\}]$  and  $\sigma_c = \sigma(c, \mathbf{Y}, \mathbf{z}) = E[\{(\mathbf{z}_i - \mu_c)^2 : y_i = c\}]$ , where  $c$  is a given class label, were considered as fundamental building blocks to model the general  $L_r$ -norm. Since mean and variance are sensitive to outliers, quartiles were added, defined as

$$\mathbf{Q}_n^j \equiv Q(j, n, \mathbf{Y}, \mathbf{z}) = \mathbf{z}_L : L = \lceil 0.25 n n_j \rceil \quad (4.9)$$

where  $\mathbf{Q}_n^j$  represents the  $n^{\text{th}}$  quartile of class  $j$ ,  $n_j$  is the number of projected samples with class label  $j$ , and  $\mathbf{z}_L$  is the  $L^{\text{th}}$  sample in the ordered set, as a robust alternative to get a summary of the dispersion and overall central tendency within each class [209]. A second element of importance for the class-information-based indices is the Rayleigh quotient, which was approximated for the 1-D case using between-class scatter  $S_B \equiv S_B(\mathbf{Y}, \mathbf{z}) = \sum_{j=1}^c (\mu_j - \mu)^2$  and within-class scatter  $S_W \equiv S_W(\mathbf{Y}, \mathbf{z}) = \sum_{j=1}^c \sum_{i=1}^{n_j} (\mathbf{z}_i - \mu_j)^2$ , appropriately included as members in the function set, where  $\mu_j$  is the mean of class  $j$ , and  $\mu$  is the global mean. An example of such class-information-based indices in terms of the proposed function and terminal sets is displayed in Fig. 4.3(a). Table 4.1a summarises the proposed set of functionals.

Three parameters are needed to fully define Rényi divergence: an estimation of the projected data pdf, the  $\rho$  coefficient and the reference pdf. To allow the evolution

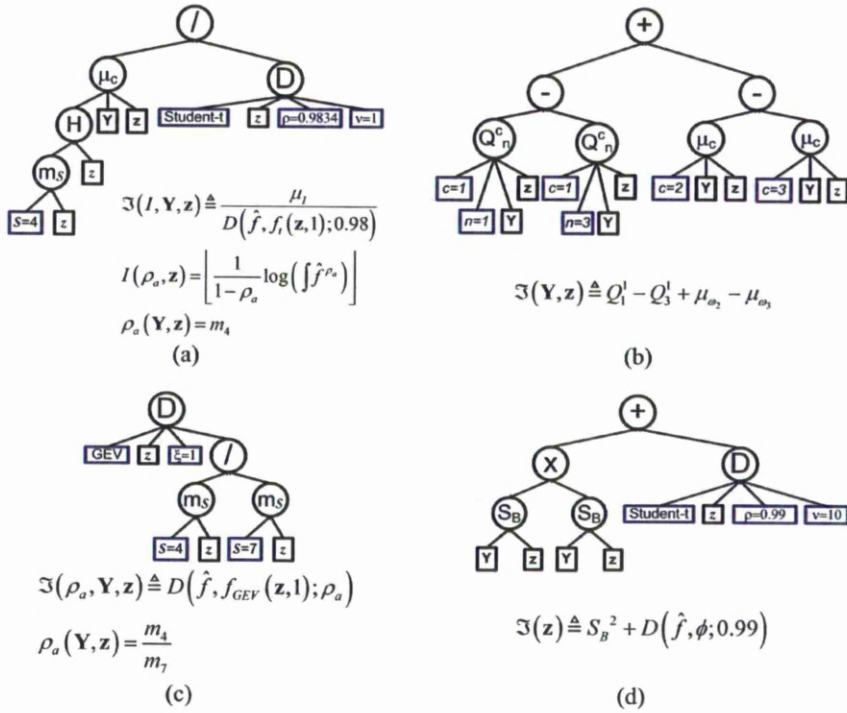


Figure 4.3: Possible evolved trees. (a) Interaction between supervised and unsupervised functions; a member of the function set is used instead of an ephemeral constant as input to another function, and it needs to be rounded by means of the floor function. (b) Pure supervised index; four supervised nodes resembling a linear combination of supervised projection indices. (c) Reference density dependant on moments; the  $\rho$  coefficient depends on the moments ratio. (d) Segregated supervised and unsupervised branches; the supervised branch (left) and the unsupervised branch (right) are independent of each other but combined through a binary operator.

process to select the right reference pdf, three additional scalar arguments for Rényi divergence are included. The first argument ( $\tau$ ) selects among the three different pdfs (i.e., Gaussian, Student-t or GEV); the second one specifies the degrees of freedom ( $\nu$ ) for the Student-t distribution; and the last one is the shape parameter ( $\xi$ ) for the GEV distribution. These arguments along with the  $\rho$  coefficient are implemented by means of the terminal  $\eta$ , which takes a random value when the potential index function is built, and then remains constant for purposes of index evaluation unless the genetic operators changed it (i.e., mutation can change the values of this ephemeral constants). The order  $s$  of the moment  $m_s$ , the quartile number  $n$ , and the class index  $c$  in the mean  $\mu_c$ , variance  $\sigma_c$  and quartile  $Q_n^c$  were implemented in a similar way. The remaining of the terminal set, summarised in Table 4.1b, was built up to accommodate the arguments needed by members of the function set. For instance, most of the functions in Table 4.1a receive as arguments the projected data  $\mathbf{z}$  with exception of arithmetic operators. For function members using class information the class labels  $\mathbf{Y}$  are required.

As was explained earlier, the genetic library defined above is adequately expressive to create most PP indices presented in Section 2.3.3.2, but also create many new ones not considered before. Figure 4.3 displays a number of examples of hypothetical but valid new indices. It includes simple cases (e.g., Fig. 4.3(d)) where the unsupervised and supervised part are clearly identifiable, as well as complex relations where the supervised part defines parameters governing the behaviour of the unsupervised part (e.g., Fig. 4.3(c), where the ratio of sample central moments defines the coefficient  $\rho$  of the Rényi divergence). This type of dependency is translated into a composition of functions, where some of the index's parameters are functions of other projection quantities. In such case, the variable parameters precede the labels and projected data in the arguments of the projection index.

## 4.4 Experimental Results

### 4.4.1 Datasets

In order to assess the generalization performance of EPP, a total of five datasets obtained from the UCI Machine Learning Repository were tested: breast cancer Wisconsin dataset (cancer) in its diagnostic variation, statlog heart dataset (heart), Pima Indians diabetes dataset (diabetes), wine dataset (wine), and glass identification dataset (glass). Categorical features were replaced with binary variables. A summary of the datasets can be found in Table 4.2.

Table 4.2: Datasets Summary.

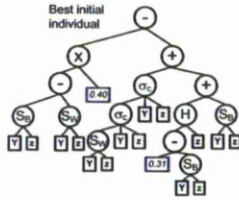
Dataset	Features	Classes	Samples
Cancer	30	2	568
Wine	13	3	178
Heart	13	2	270
Glass	9	3	214
Diabetes	8	2	768

### 4.4.2 Evolution Process

The GP algorithm was run over 50 generations with 60 individuals in the population; the crossover and mutation rate were managed dynamically by GPLAB [210], which was also the library used to implement the GP algorithm. Controls against bloat (e.g. dynamic depth/size on the individual trees) were also activated at the beginning of each GP trial. All experiments were run on a PC with CPU Intel Pentium 4 at 3.08 GHz, 1.00 GB in RAM and with Microsoft Windows XP Professional SP2 as operating system; MATLAB 2009a was used to implement the required algorithms.

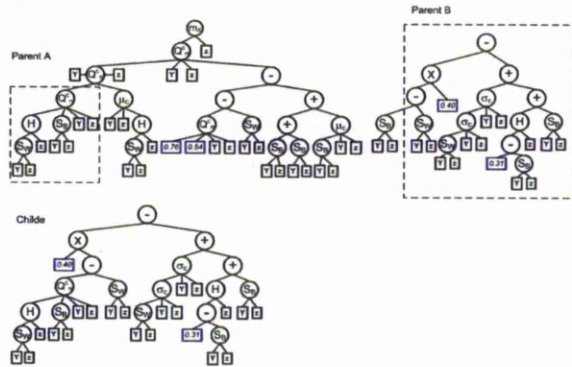


Generation 1



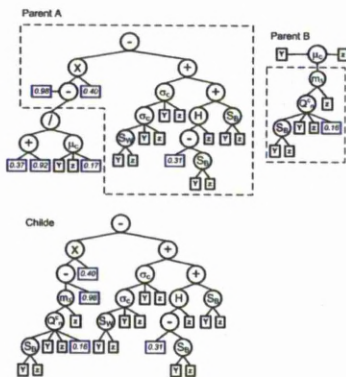
Fitness: 0.2063

Generation 5



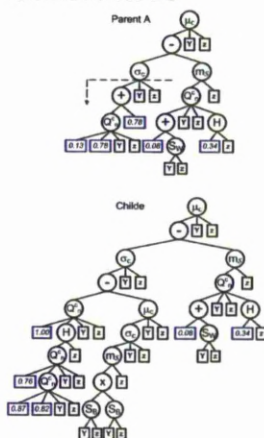
Fitness: 0.1809

Generation 15



Fitness: 0.1037

Generation 20



Fitness: 0.0809

Figure 4.4: Induction process for wine dataset. Selected generations showing the best individual, along with its parents and the genetic operator selected to build it. Its fitness function is given at the bottom of each best individual.

Figure 4.5 shows a sample of the fitness function minimisation carried out by the proposed evolution process. Best so far, population fitness average and standard deviation are displayed. It can be seen that convergence is reached after generation 30. An example of the induction process modelled by Eq. (4.1) can be observed in Fig. 4.4, where the best individual in the population is displayed as a tree representation, along with its corresponding fitness value. To illustrate how the evolution process helps to infer a suitable projection index, four different generations were selected from the whole evolution process. In each case, the parents of the best member in the population and the genetic operator involved in its making are shown. When crossover is selected as genetic operator, the crossover point in every parent is highlighted, when mutation is selected, the mutation point is signalled by an arrow, indicating the fraction of the tree to be removed.

Table 4.3: Comparison of the proposed algorithm including median values and interquartile ranges of error assessed with 10-CV, optimal classifier parameters for existing indices, average times taken for feature extraction and classification of a single fold ( $T_1$  in seconds), and times taken to evolve a single index ( $T_2$  in hours).

Friedman & Tukey $\mathfrak{S}_1$						Jones & Sibson $\mathfrak{S}_3$					
Problem	Median	IQR	$[k, \lambda]$	$N_{feat}$	$T_1$	Median	IQR	$[k, \lambda]$	$N_{feat}$	$T_1$	
Cancer	6.14	7.17	[4.4, 4.8]	4.0	3.60	5.26	7.02	[3.4, 5.6]	4.0	3.56	
Wine	19.44	11.11	[5.4, 5.9]	3.0	1.99	22.22	16.67	[6.5, 3.9]	2.9	1.04	
Heart	48.15	22.22	[5.5, 5.7]	11.0	0.99	31.48	22.22	[7.6, 6.1]	7.1	1.45	
Glass	35.71	20.56	[3.0, 3.8]	5.4	2.91	37.23	7.58	[3.2, 5.4]	5.3	3.92	
Diabetes	31.37	8.22	[7.6, 5.5]	5.6	4.80	36.61	7.79	[7.2, 5.1]	4.6	5.67	
Information Divergence $\mathfrak{S}_4$						Fisher Information $\mathfrak{S}_2$					
Problem	Median	IQR	$[k, \lambda]$	$N_{feat}$	$T_1$	Median	IQR	$[k, \lambda]$	$N_{feat}$	$T_1$	
Cancer	5.51	7.02	[4.8, 5.1]	4.0	2.54	5.51	8.96	[5.2, 4.4]	4.0	3.76	
Wine	16.67	5.56	[6.9, 3.7]	2.9	1.08	25.00	22.22	[5.2, 4.7]	2.5	1.29	
Heart	33.33	11.11	[5.7, 5.3]	5.4	1.44	29.63	11.11	[6.9, 5.4]	4.2	1.68	
Glass	30.19	19.05	[3.0, 5.5]	4.7	2.88	38.10	22.73	[4.0, 5.4]	5.6	4.71	
Diabetes	32.47	8.24	[8.7, 6.6]	5.0	3.57	30.72	9.09	[7.5, 6.4]	4.1	5.98	
Skewness $\mathfrak{S}_5$						Kurtosis $\mathfrak{S}_6$					
Problem	Median	IQR	$[k, \lambda]$	$N_{feat}$	$T_1$	Median	IQR	$[k, \lambda]$	$N_{feat}$	$T_1$	
Cancer	7.06	7.02	[4.4, 2.7]	4.0	1.06	5.26	7.17	[4.0, 2.5]	4.0	0.93	
Wine	27.78	16.67	[5.2, 8.7]	2.9	0.13	22.22	11.11	[6.3, 4.1]	3.0	0.14	
Heart	40.74	14.81	[6.1, 5.0]	6.3	0.24	33.33	11.11	[6.1, 6.1]	10.3	0.22	
Glass	39.50	9.52	[3.2, 4.8]	5.8	0.38	39.50	19.05	[3.8, 3.4]	5.5	0.19	
Diabetes	32.02	7.79	[7.6, 3.8]	5.4	0.99	32.47	4.82	[8.5, 3.8]	5.5	1.48	
Moment Linear Combination $\mathfrak{S}_8$						Evolved Index					
Problem	Median	IQR	$[k, \lambda]$	$N_{feat}$	$T_1$	Median	IQR	$[k, \lambda]$	$N_{feat}$	$T_1$	$T_2$
Cancer	7.02	5.26	[4.2, 5.0]	4.0	1.12	4.39	1.75	[6.0, 4.90]	4.0	6.30	10.12
Wine	33.33	27.78	[6.4, 3.7]	2.7	0.16	5.56	11.11	[6.9, 5.42]	3.2	1.72	7.82
Heart	29.63	11.11	[6.6, 4.3]	7.8	0.27	18.52	11.11	[7.9, 5.72]	9.7	0.55	2.61
Glass	36.36	9.52	[2.7, 3.5]	4.9	0.31	16.67	9.52	[2.3, 5.47]	5.0	9.55	22.39
Diabetes	33.12	11.31	[6.8, 4.1]	5.3	1.55	27.92	7.36	[7.4, 5.98]	4.0	2.70	7.34
ICA						PCA					
Problem	Median	IQR	$[k, \lambda]$	$N_{feat}$	$T_1$	Median	IQR	$[k, \lambda]$	$N_{feat}$	$T_1$	
Cancer	6.14	5.20	[4.3, 2.8]	4.0	1.54	6.14	5.26	[4.0, 2.9]	6.14	1.37	
Wine	16.67	11.11	[7.3, 2.6]	3.0	0.75	22.22	27.45	[4.9, 3.0]	22.22	0.14	
Heart	20.37	3.70	[5.8, 4.0]	9.1	0.29	19.58	14.81	[6.5, 2.6]	19.58	0.28	
Glass	31.82	9.52	[2.4, 2.6]	7.0	0.28	25.54	9.52	[2.7, 1.3]	25.54	0.25	
Diabetes	30.07	6.49	[6.8, 3.8]	6.0	2.49	28.27	22.81	[7.1, 4.3]	28.27	2.46	
Fisher Linear Discriminant $\mathfrak{S}_9$						Bhattacharyya Distance $\mathfrak{S}_{10}$					
Problem	Median	IQR	$[k, \lambda]$	$N_{feat}$	$T_1$	Median	IQR	$[k, \lambda]$	$N_{feat}$	$T_1$	
Cancer	4.51	8.96	[4.4, 3.2]	4.0	0.88	4.51	7.17	[4.2, 4.5]	4.0	0.99	
Wine	19.44	6.86	[5.5, 2.4]	3.0	0.29	19.44	16.67	[6.3, 4.9]	2.7	0.51	
Heart	19.54	11.11	[7.0, 4.9]	11.0	0.23	20.37	14.81	[6.0, 4.0]	11.0	0.31	
Glass	32.58	9.52	[3.5, 4.9]	7.0	0.30	31.82	14.29	[2.3, 7.5]	6.1	0.43	
Diabetes	28.62	5.19	[7.6, 5.8]	6.0	1.44	28.62	3.90	[8.5, 7.1]	6.0	1.52	

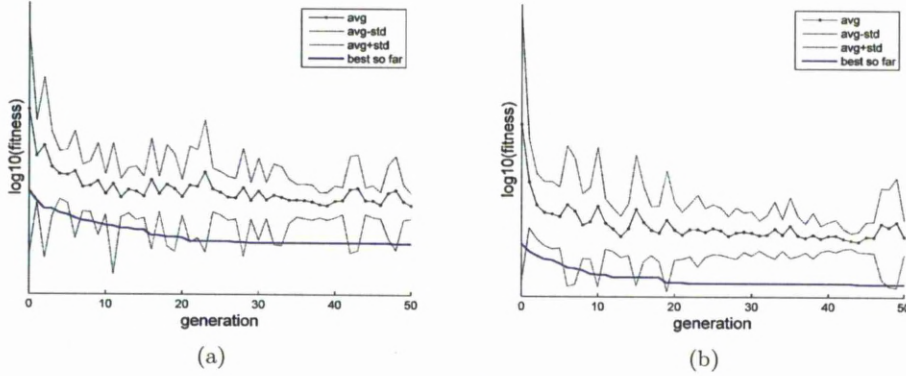


Figure 4.5: Minimisation of fitness function by the proposed evolution process for (a) cancer and (b) heart.

#### 4.4.3 Training and Testing Set Design

To objectively evaluate the performance of the proposed system, and at the same time optimise the best feature extraction/classification pair, a 10-CV partition scheme was used for final model assessment. Specifically, in each fold, 90% of the dataset was used for the learning phase (i.e. the input data  $\Upsilon = \{\Omega \cup \bar{\Omega}\}$  to the proposed system), and the remaining 10% samples  $\bar{\Upsilon}$  were used to test the generalization performance of the proposed system. Only the data within the learning phase is used to perform the model selection, via the 3-CV and the genetic search, which returns the optimal index (tree) and kNN parameters  $k$  and  $\lambda$ . The entire model selection is repeated within each fold of the model assessment 10CV procedure. Descriptive statistics over the ten folds are reported in Table 4.4.

#### 4.4.4 Baseline Projection Pursuit Comparison

This section compares the accuracy of the features extracted using the proposed evolutionary framework with the ones obtained from existing indices used in PP, including Friedman and Tukey (F&T)  $\mathfrak{S}_1$ , Jones and Sibson (J&S)  $\mathfrak{S}_3$ , Fisher information  $\mathfrak{S}_2$ , information divergence  $\mathfrak{S}_4$ , skewness  $\mathfrak{S}_5$ , kurtosis  $\mathfrak{S}_6$ , moments linear combination  $\mathfrak{S}_8$ ,  $L_r$ -norm  $\mathfrak{S}_9$  with  $r = 2$  (FLD), and Bhattacharyya distance  $\mathfrak{S}_{10}$ . Additionally, this comparison includes the features extracted with standard unsupervised feature extraction methods like PCA and ICA, as given in [50] and [49], respectively.

To make this comparison fair, the same partitioning scheme was used. For indices using the projected data pdf, an estimation was provided by means of a normal kernel using a window parameter that is a function of the number of points. Since  $\mathfrak{S}$  was given, the remaining unknown parameter in the proposed model  $\theta = \{k, \lambda\}$  was selected by performing a grid search using  $\Upsilon$  to train the classifier. After the optimal  $\theta$  was found,

Table 4.4: Summary of Table 4.3, where each cell displays the median of the 10-CV error along with the average number of features found by the Bayesian stopping criterion.

Index	Dataset (Median error / # Features)					
	Cancer	Wine	Heart	Glass	Diabetes	Diabetes
ICA	6.14/4.0	16.67/3.0	20.37/9.1	31.82/7.0	30.07/6.0	30.07/6.0
PCA	6.14/4.0	22.22/3.0	19.58/11.0	25.54/7.0	28.27/3.0	28.27/3.0
S <sub>1</sub> : Friedman & Tukey	6.14/4.0	19.44/3.0	48.15/11.0	35.71/5.4	31.37/5.6	31.37/5.6
S <sub>3</sub> : Jones & Sibson	5.26/4.0	22.22/2.9	31.48/7.1	37.23/5.3	36.61/4.6	36.61/4.6
S <sub>4</sub> : Information Divergence	5.51/4.0	16.67/2.9	33.33/5.4	30.19/4.7	32.47/5.0	32.47/5.0
S <sub>2</sub> : Fisher information	5.51/4.0	25.00/2.5	29.63/4.2	38.10/5.6	30.72/4.1	30.72/4.1
S <sub>5</sub> : Skewness	7.06/4.0	27.78/2.9	40.74/6.3	39.50/5.8	32.02/5.4	32.02/5.4
S <sub>6</sub> : Kurtosis	5.26/4.0	22.22/3.0	33.33/10.3	39.50/5.5	32.47/5.5	32.47/5.5
S <sub>8</sub> : Moments linear combination	7.02/4.0	33.33/2.7	29.63/7.8	36.36/4.9	33.12/5.3	33.12/5.3
S <sub>9</sub> : Fisher linear discriminant	4.51/4.0	19.44/3.0	19.54/11.0	32.58/7.0	28.62/6.0	28.62/6.0
S <sub>10</sub> : Bhattacharyya distance	4.51/4.0	19.44/2.7	20.37/11.0	31.82/6.1	28.62/6.0	28.62/6.0
Evolved index	<b>4.39/4.0</b>	<b>5.56/3.2</b>	<b>18.52/9.7</b>	<b>16.67/5.0</b>	<b>27.92/4.0</b>	<b>27.92/4.0</b>

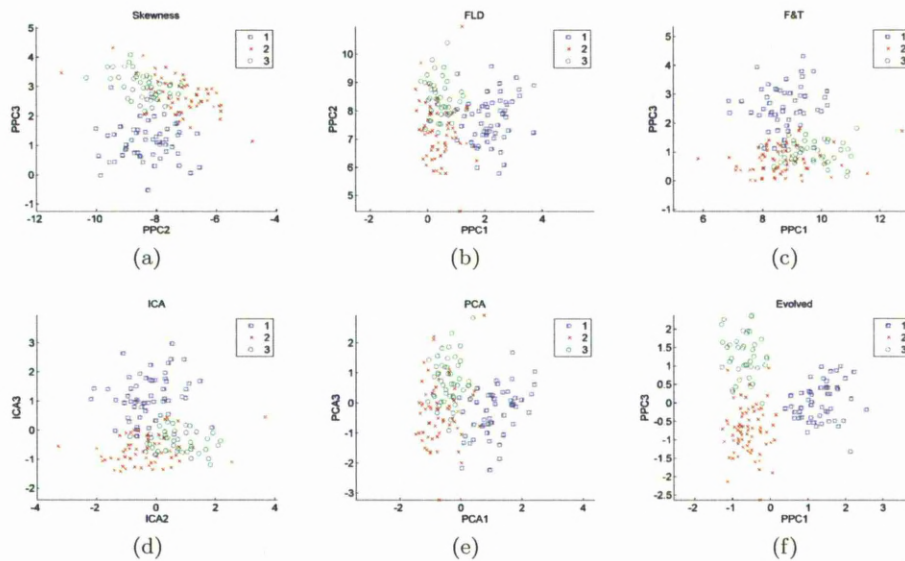


Figure 4.6: 2D scatter plots of the best two projected features for wine data using (a) skewness, (b) FDA and (c) F&T projection indices for feature extraction; contrasting with scatter plots using (d) ICA, (e) PCA and (f) EPP

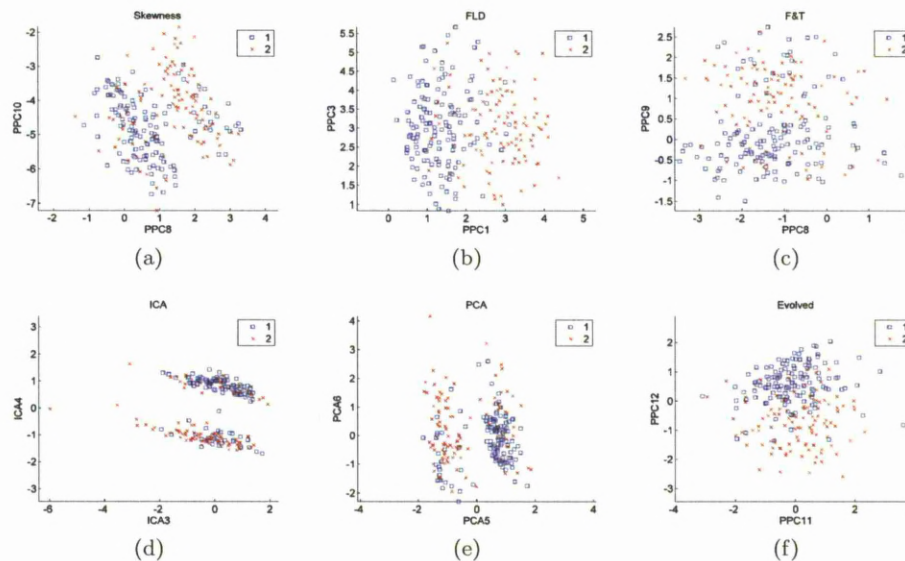


Figure 4.7: 2D scatter plots of the best two projected features for heart data using (a) skewness, (b) FDA and (c) F&T projection indices for feature extraction; contrasting with scatter plots using (d) ICA, (e) PCA and (f) EPP

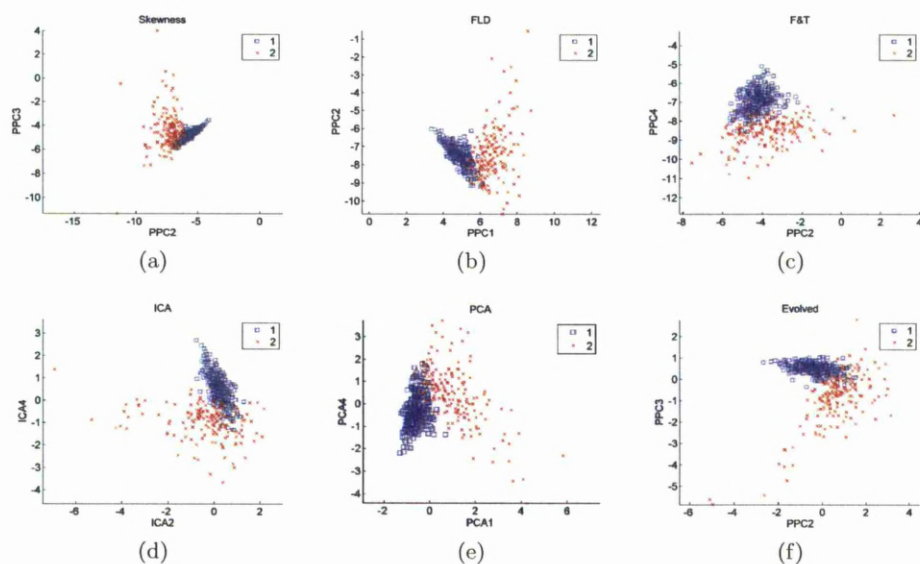


Figure 4.8: 2D scatter plots of the best two projected features for cancer data using (a) skewness, (b) FDA and (c) F&T projection indices for feature extraction; contrasting with scatter plots using (d) ICA, (e) PCA and (f) EPP

the projected samples in  $\bar{\mathbf{Y}}$  were classified and the median of the classification errors over the ten folds is reported as the generalization performance in Table 4.3, along with the interquartile range (IQR), the average of the optimal classifier parameters, the average number of dimensions and the time taken to perform feature extraction and classification with the corresponding projection index for a single fold.

Table 4.4 summarizes and compares the 10-CV performance for the evolved index and those eleven existing indices, from which it can be seen the evolved index performed the best. As it was expected, classical supervised indices performed better, on average, than unsupervised indices, frequently ranking in second or third place. Among the two supervised indices, Bhattacharyya distance showed to be more consistent, as it was ranked on third place for most of the classification problems considered on Table 4.3. As for the unsupervised indices, PCA performed on average equally good as Bhattacharyya distance, although PCA showed to be less consistent as it showed a variance of almost double than  $\mathfrak{S}_{10}$ . Surprisingly, a big difference can be observed between the performance of ICA and indices based on high-order statistics, such as kurtosis, skewness, and moments linear combination. Finally, the reader can observe that the worst possible choice of projection index for the selected problems is  $\mathfrak{S}_6$  because, although it is not ranked last in every problem, it does not exhibit a consistent behaviour.

Besides classification error, Table 4.3 also displays the average number of dimensions induced for each feature-extraction/classifier pair. Although the evolved indices did not always deliver the minimum number of extracted features, for example in the wine

dataset the evolved index extracted the maximum number of features when compared to the others, they do extract informative features as it is confirmed by their classification performance. As for the classifier hyperparameters, also showed in Table 4.3 for each index in this comparison, the number of nearest neighbours induced by EPP is within one and a half standard deviations from the mean of the distribution formed by this same parameter induced for the rest of the indices. This could indicate the parameters induced by the proposed evolutionary system could be trusted at 95% of confidence. The compression power of PCA can also be observed in Table 4.3, where on average PCA ranked first among the selected indices when measuring the optimal induced number of neighbours needed for classification. This characteristic together with the low dimensionality of the extracted features show why PCA is highly prone to loss of discriminatory information. On the contrary, although the evolved indices can not extract such low dimensional spaces, they provide reliable and optimal spaces with maximum class separability.

To illustrate the classification ability of the extracted features with EPP, 2-D scatter plots of selected features are compared for six different indices (i.e., skewness, FLD, F&T, ICA, PCA, and evolved index), for wine, heart, and cancer datasets in Figs. 4.6-4.8. It can be seen the evolved index present better class separability, as can be confirmed from the results displayed in Table 4.3. Additionally, a sensitivity analysis over the extracted features is presented for the heart dataset using the tools provided by Cardillo [211]. Receiver operating characteristic (ROC) curves [42] were computed for both features used to build the scatter plots in Fig. 4.7, and the ROC curve for the dominant feature for each index is displayed in Fig. 4.10 along with the area under ROC curves ( $A_z$ ). From this figure it is clear the extracted features using the evolutionary approach (and also the FDA for this dataset) have superior discrimination power. As expected, the features extracted with the indices involving entropy and its approximation displayed lower class discrimination, being overcome by supervised indices.

Considering that GP produces trees that are often difficult to intuitively interpret, a simple synthetic dataset is used to demonstrate the generation of a simple tree and contrast it with the intuitively defined PCA and ICA indices. The synthetic dataset was built from two 2-D elongated Gaussians of identical covariances, and positioned as shown in Fig. 4.9. To provide a fair comparison with ICA and PCA, the supervised members of the function set were removed. The smallest evolved tree/index, shown in Fig. 4.9.d, is an instance of the Rényi entropy for a specific  $\rho$  and some offset. This is expected as the maximum value for Rényi entropy is achieved when the projected data distribution deviates from a Gaussian. This is quantitatively confirmed by the 1-D projection axis, which matches closely the one by ICA, while PCA as expected fails to locate a discriminatory projection. Fig. 4.9.b and 4.9.c summarize the differences of

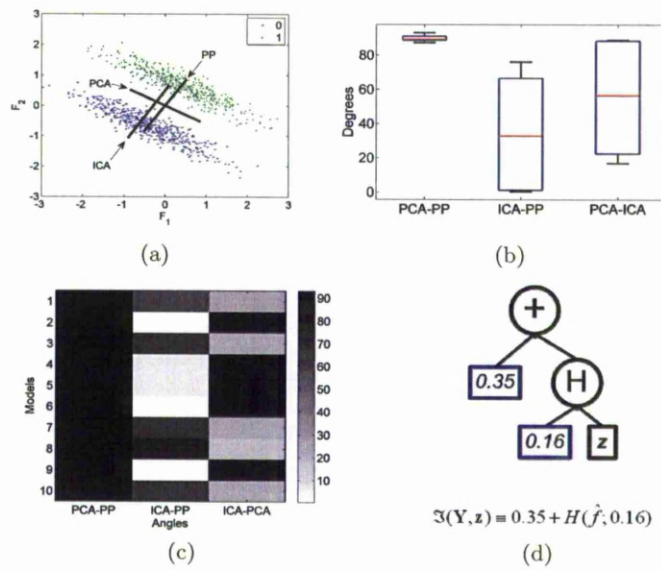


Figure 4.9: Synthetic dataset analysis. (a) Projected vectors extracted using PCA, ICA, and the proposed method, labeled as PP. (b) Boxplots and (c) distance matrix showing the angle distribution between the axis of every compared method. (d) Smallest optimal generated tree.

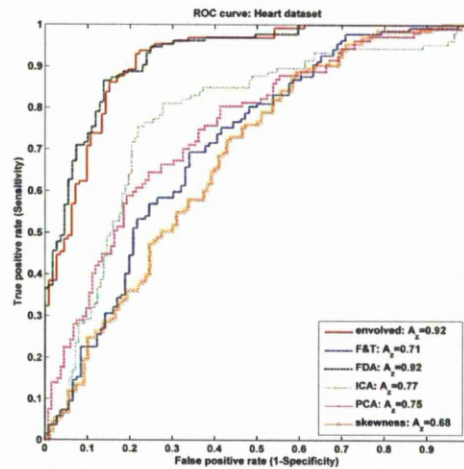


Figure 4.10: Examples of ROC curves using the best projection methods for heart dataset.



the axes between all pairs of methods for each model generated in each 10-CV fold. As can be seen, the PP generated axes are close with those generated by ICA but are at near 90° to those generated by PCA. Interestingly, as shown in Fig. 4.9.b and 4.9.c, PP proved to be more stable than ICA, when both methods are compared against PCA as ground truth.

#### 4.4.5 Comparison of EPP with Collaborative Methods

Given the ability of the proposed system to generate a complex composition of functionals, and thus a combination of the existing indices, it is natural to compare its performance against other methods for combining projection based classifiers. Three methods were selected to compete against the inferred indices, namely linear weighted sum, majority vote [212] and projection onto convex sets [213]. The first method was selected to demonstrate the difference between the evolution of projection indices, and a simple linear combination of the six most common indices from Tables 2.1 and 2.2, described by

$$\mathfrak{S}_L = \sum_{k=1}^6 \omega_k \mathfrak{S}_k(\mathbf{X} \cdot \mathbf{p}_i). \quad (4.10)$$

Each weight in Eq. (4.10) is learnt from the data and modelled as a system hyperparameter. This process was implemented following the same partition framework as described in Section 4.4.3. The performance of a kNN classifier using the extracted features resulting from projection pursuit with  $\mathfrak{S}_L$  as projection index, is reported in Table 4.5. Clearly, although  $\mathfrak{S}_L$  is capable of varying the contribution of each projection index through its corresponding weight, it is not flexible enough to explain the underlying structure in the data as well as the newly evolved projection indices.

Majority vote was selected because of the advantages of the classifier ensembles [212]. It is well known that an ensemble method can improve the performance of otherwise weak set of classifiers, therefore the evolved indices are compared against a classifier ensemble composed of six individual projection indices,  $\mathfrak{S}_1$ - $\mathfrak{S}_6$  in Tables 2.1-2.2, each one concatenated with a single kNN classifier, thus giving a ensemble of six classifiers. The parameters of each individual classifier are considered system hyperparameters using the training, testing, and validation scheme proposed in Section 4.4.3. The out-of- sample classification error estimated with 10-CV is reported in Table 4.5 along with the mean of the estimated classifier parameters. Surprisingly, the ensemble performance is not always better than the linear weighted sum index, although the low IQR values indicate congruency among the individual classifiers, making the ensemble robust to sampling of the training set.

Projection onto convex sets (POCS) is an iterative algorithm aimed at solving optimisation problems whose solution lies in the intersection of spaces defined by the problem constraints. These sets are required to be convex so that the method converges.

Table 4.5: Contrasting collaborating methods. Median and interquartile range of 10-CV error, together with the mean optimal classifier parameters.

Problem	Weighted Sum				POCS			
	Median	IQR	$[k, \lambda]$	Features	Median	IQR	$[k, \lambda]$	Features
Cancer	8.77	8.77	[6.0, 4.7]	2.5	5.39	7.02	[2.5, 3.9]	97
Wine	16.67	6.86	[5.5, 7.7]	3.0	16.67	6.54	[3.0, 5.6]	34
Heart	27.78	11.11	[7.0, 6.3]	4.0	22.22	18.52	[6.5, 4.6]	35
Glass	34.85	14.29	[3.0, 7.0]	3.5	23.81	8.23	[1.0, 1.6]	24
Diabetes	29.41	6.90	[6.0, 2.7]	4.0	32.54	7.35	[5.1, 7.6]	30

Majority Vote														
Problem	Median	IQR	C1 $[k, \lambda]$	Feat	C2 $[k, \lambda]$	Feat	C3 $[k, \lambda]$	Feat	C4 $[k, \lambda]$	Feat	C5 $[k, \lambda]$	Feat	C6 $[k, \lambda]$	Feat
Cancer	6.14	7.02	[3.7, 6.0]	4.0	[5.1, 2.9]	4.0	[4.2, 4.7]	3.3	[5.0, 4.2]	3.4	[3.6, 4.8]	3.5	[4.4, 4.9]	2.5
Wine	22.22	1.31	[4.8, 5.5]	2.9	[5.3, 4.3]	2.7	[5.4, 5.7]	2.8	[5.5, 4.7]	3.0	[4.5, 4.7]	2.9	[4.4, 5.5]	2.7
Heart	28.57	4.32	[3.0, 5.2]	4.1	[4.3, 3.3]	3.7	[5.2, 5.3]	3.8	[3.7, 4.3]	4.0	[4.3, 5.8]	3.2	[3.2, 5.7]	4.2
Glass	28.57	6.06	[2.6, 3.7]	3.0	[2.7, 6.2]	3.2	[2.7, 5.5]	3.7	[3.5, 4.3]	3.0	[4.1, 4.3]	3.8	[3.1, 5.7]	4.5
Diabetes	67.97	7.79	[3.2, 1.7]	2.6	[4.5, 4.3]	2.9	[2.7, 4.6]	2.8	[4.4, 2.7]	4.0	[4.7, 5.9]	3.3	[3.9, 3.3]	4.7

POCS makes use of predefined projection operators on a single set to successively project a starting point from one convex set to another until it falls inside the solution area. The most frequent applications of POCS are image restoration, denoising and recently super-resolution (SR) image reconstruction from low-resolution (LR) samples [7]. For the purpose of comparison, POCS was implemented as in [214] assuming each sample was a LR sample image of an unknown high resolution version. Taking each sample as the reference image and its nearest neighbors within the same class as LR versions, POCS is applied to construct a SR image for each sample in the dataset. This process embeds the dataset into a high-dimensional space where each point is representative of its within-class nearest neighbors in the original feature space. Then PCA is applied to reduce the number of dimensions and facilitate the classifier task. After this preprocessing stage, the extracted features are used to infer the parameters of a kNN classifier using the proposed partition scheme in Section 4.4.3. The out-of-sample estimation error of a 10-CV is reported in Table 4.5 along with the mean of the inferred classifier parameters and the number of used features. In general, POCS performs better than the other two collaborated schemes, nevertheless the results show the evolved indices outperform any of the collaborated schemes considered.

#### 4.4.6 Evolved Indices as a Tree Representation

Finally, Fig. 4.11 presents the best evolved index for each dataset. It is worthy to highlight from this figure the fact that in each of the evolved indices at least one term considers class information, which suggests that, as expected, that class information plays an important part in feature extraction for classification. Additionally, Rényi entropy and Rényi divergence did not appear in the same tree for all the datasets, which indicates they may have overlapping information extraction properties.

The most complex index obtained was the one for the glass dataset, Fig. 4.11(e), which exhibits a highly unbalanced structure with the highest number of unsupervised nodes and a depth of nine levels. Contrastingly, the evolved index for the cancer dataset, Fig. 4.11(a), presents a fairly balanced structure and a depth of four levels. Despite the inclusion of unsupervised nodes in the structure of all the evolved indices, the ratio between supervised and unsupervised nodes leans toward the supervised side for most of the indices. The advantage of a highly supervised index, like the one in Fig. 4.11(b), can be seen from Fig. 4.11(f) where the evolved index presented the best class separability among the compared indices.

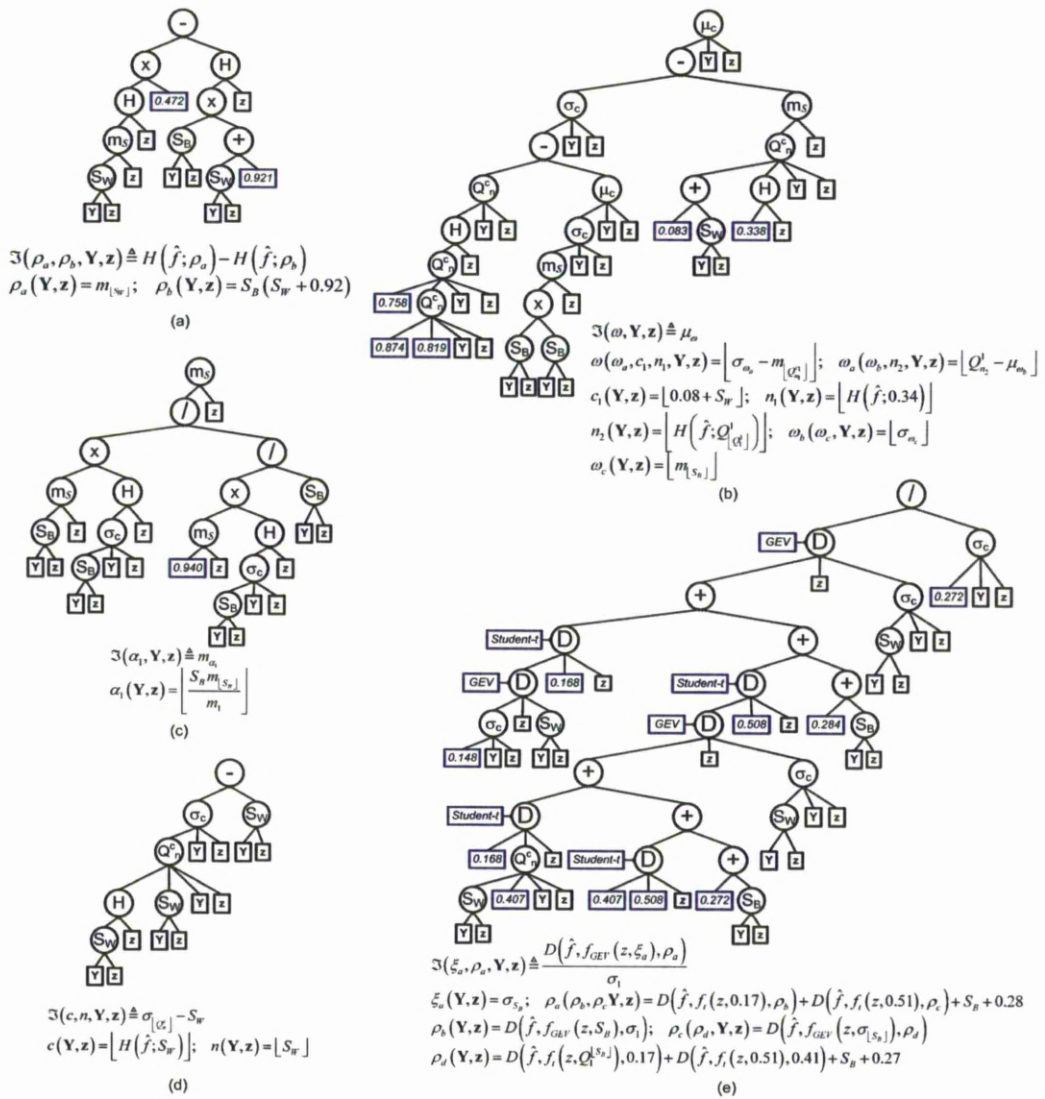


Figure 4.11: Best evolved trees for different datasets. (a) Cancer: a highly symmetric tree where the parameters of the unsupervised functions are defined by supervised nodes. (b) Wine: a highly unbalanced tree mainly built with supervised nodes. (c) Heart: a compact tree with redundant branches (Rényi entropy on the left and right of the first division node), but good generalization performance. (d) Diabetes: unbalanced and highly supervised tree where within-class scatter plays an important role. (e) Glass: very complex and unbalanced tree with equal number of supervised and unsupervised nodes.

## 4.5 Summary

In this chapter an evolutionary framework for automatic design of classification systems was introduced. It poses the learning problem as a model selection problem, consisting of two stages. In the first stage, a set of potential models is built with the help of genetic operators, then in a second stage they are evaluated using a wrapper approach, where cross-validation is used to estimate the out-of-sample classification error of each potential model. After a given number of generations, the fittest individual is chosen as the desired model, guaranteeing optimal classification performance. In this work, a classification system is modelled as a pair consisting of a feature extraction stage and a classification stage. Due to the strong interactions between classifier and feature extraction, both stages need to be jointly optimised, thus this work proposes to encode the feature-extraction/classifier pair into a single hybrid chromosome designed specifically to hold a tree structure.

As a starting point and due to its simple, yet powerful, formulation, this chapter also discussed the advantages of the proposed evolutionary framework when the feature extraction stage is modelled as a linear projection method. It was showed that the proposed inducer evolves ad-hoc PP indices which extract highly discriminative features, out performing those extracted with several existing indices and popular linear feature extraction methods with close analytical solutions, such as PCA, ICA and FLD. A second advantage of the proposed system is its degree of precision, which was measured via the IQR of the 10 folds used for final model assessment. Such range showed to be minimum for the evolved indices when compared to other projection indices for five given classification problems as can be seen in Table 4.3. Additionally to measuring the precision of the evolutionary framework in producing accurate classifiers, the accuracy of the extracted features via PP was measured in the experiment illustrated in Figure 4.9. It was showed that an evolved index produces consistent features more frequent than ICA, which also uses an iterative method to compute independent features.

Additionally, this chapter presented a comparison between EPP and three collaborative feature extraction methods. The first collaborative method was a PP index designed to be the weighted sum of six existing projection indices, which weights were jointly optimised with a GA. The second collaborative method was POCS [214], which represents a different paradigm of projection methods to solve optimisation problems. Finally, the same PP indices used in the weighted sum were combined using majority vote, which individually trains each classifier and then assigns the most frequent label predicted by the individual classifiers. Although POCS exhibited on average the lowest classification error among the collaborative methods, it could not outperform EPP. Linear weighted sum came second, showing lower dimensionality spaces.

As with all the evolutionary optimisers, the proposed system exhibit long learning curves, therefore it can not be used for online learning. Nevertheless, once the proposed

evolutionary framework produces an evolved index, it is guaranteed to deliver accurate and precise features with high discriminative capabilities. Additionally, It has been experimentally illustrated that EPP successfully performs model selection in the space of potential indices to be used in SPP for optimal feature selection. Finally, it is worth to mention that if the data is not linearly separable in the original feature space, neither will be in the space described by the features extracted using linear projection techniques. This situation is commonly faced for classification problems involving real-world, high-dimensional datasets as discussed in Section 2.3.5. Classical linear feature extraction methods are extended to tackle nonlinearities in the dataset by projecting the data into a non-observable feature space to unfold undesired nonlinearities, as explained in Sections 2.3.2. In the next chapter we elaborate on such extension for EPP.

## Chapter 5

# Nonlinear Projection Pursuit via Kernel-Induced Spaces

### 5.1 Introduction

The previous chapter explored PP as a generalization of linear projection techniques, and developed an evolutionary framework that delivers a feature-extraction/classifier pair modelled as a projection index  $\mathfrak{S}$  and a set of hyperparameters  $\theta = \{k, \lambda\}$ , tailored to a given classification problem. A natural enhancement to the generalization abilities of PP is its extension to the nonlinear case. The first reference to such idea can be traced back to [215] where an exploratory method was proposed to investigate nonlinear structure based on Hebbian learning applied to train a neural network. Later on, Smola et al. [216] proposed to learn a set of projections by optimising a constant function (index) in a kernel-induced feature space, chosen from variance, Fisher information, negative Shannon entropy, or other quantities of interests, called kernel PP. Following the guidelines listed by Friedman [217] for PP, kernel PP also discusses two main possible choices on structure removal process: (1) by removing from the search space the previously obtained projections; (2) by applying Gram-Schmidt orthonormalization in the kernel-induced feature space.

Facing a specific classification problem, prior information and specific data distributions have to be considered in the index selection for PP. However, it is not easy to achieve this in the non-observable kernel space, where the data distribution may be distorted and is untraceable. To overcome these problems, a variation of the evolutionary framework previously presented is introduced in this chapter, aiming at extraction of nonlinear features in the kernel-induced feature space. The development of a structure removal process can be found among the main contributions of this chapter. Such removal process is different from those used in [216], since it reduces the determination of the nonlinear residual subspace to the computation of an updated kernel matrix. Additionally, analysis on the kernel-based whitening process is also provided. To test the effectiveness of the proposed approach, final model assessment is performed over

six different high-dimensional datasets via 10-fold cross-validation (10-CV).

## 5.2 Problem Formulation in Kernel-Induced Spaces

Based on the Mercer's theorem [218], a kernel function defines the dot product in a non-observable feature space  $\mathcal{H}$ , which is called kernel-induced feature space. Letting  $\phi : \mathbb{R}^m \rightarrow \mathcal{H}$  denote the nonlinear mapping to  $\mathcal{H}$ , and  $\mathbf{K} = [k_{ij}]$  denote the kernel matrix between the data points, the entry  $ij$  of  $\mathbf{K}$  is known as  $k_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)\phi^T(\mathbf{x}_j)$ . Working in this new space, a projection vector in  $\mathcal{H}$  is sought so that the projected features possess the maximum degree of interestingness:

$$\boldsymbol{\nu}^* = \operatorname{argmax}_{\boldsymbol{\nu} \in \mathcal{H}} \mathfrak{S}(\Phi\boldsymbol{\nu}), \quad (5.1)$$

where  $\Phi = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \dots, \phi(\mathbf{x}_i), \dots]^T$  represents the mapped data in  $\mathcal{H}$ . Since the mapping  $\phi$  is unknown, it is impossible to directly compute such projection vector. However, by approximating this non-observable kernel space  $\mathcal{H}$  by a subspace spanned by a set of data points from the training set, the projection vector can be expressed as

$$\boldsymbol{\nu} = \Phi^T \boldsymbol{\gamma}, \quad (5.2)$$

where  $\boldsymbol{\gamma} \in \mathbb{R}^n$  is a set of coefficients defining the basis of the subspace. Thus, in the kernel-induced feature space, the projected features of the mapped data  $\Phi$  onto  $\boldsymbol{\nu} \in \mathcal{H}$  can be computed by

$$\mathbf{Z}(\boldsymbol{\gamma}) = \Phi\boldsymbol{\nu} = \mathbf{K}\boldsymbol{\gamma}. \quad (5.3)$$

and the constraint in Eq. (2.34) turns into  $\Gamma^T \mathbf{K} \boldsymbol{\gamma} = \mathbf{I}$ .

Hence, for the standard PP optimisation, the optimal coefficient vector is computed, instead of the optimal projection vector in Eq. (5.1), by solving the following optimisation problem:

$$\boldsymbol{\gamma}^* = \operatorname{argmax}_{\boldsymbol{\gamma} \in \mathbb{R}^n} \mathfrak{S}(\mathbf{K}\boldsymbol{\gamma}). \quad (5.4)$$

Consequently, to obtain an optimal index/classifier pair in the kernel-induced feature space, the following bi-level optimisation problem is solved:

$$\begin{aligned} \min \quad & F(\mathfrak{S}, k, \lambda; \mathbf{K}) = \sum_{i=1}^h \sum_{j \in \operatorname{Ind}(\bar{\Omega}_i)} L(y_j, \varphi_{\Omega_i}(\bar{\mathbf{K}}_{ji}; \boldsymbol{\gamma}^*, \boldsymbol{\theta})) \\ \text{s.t.} \quad & \boldsymbol{\gamma}^* = \operatorname{argmax}_{\boldsymbol{\gamma} \in \mathbb{R}^n} \{\mathfrak{S}(\mathbf{K}_i \boldsymbol{\gamma})\} \end{aligned} \quad (5.5)$$

where  $\operatorname{Ind}(\bar{\Omega}_i)$  denotes the sample indices in the validation set  $\bar{\Omega}_i$ ,  $\mathbf{K}_i$  denotes the kernel matrix between samples from the training set  $\Omega_i$ ,  $\bar{\mathbf{K}}_{ji}$  denotes the kernel matrix between the  $j^{\text{th}}$  sample from the validation set  $\bar{\Omega}_i$  and all the samples from the training set  $\Omega_i$ , and  $\mathbf{K}$  here denotes the kernel matrix between samples from  $\Upsilon = \Omega_i \cup \bar{\Omega}_i$ . It



can be seen from Eq. (5.5), the used evolutionary framework in the kernel-induced feature space are expressed only in terms of the kernel matrix.

### 5.3 Nonlinear Sequential Removal Process

Often, a single direction is not enough to represent the underlying data structure, thus PP needs to be iterated to obtain a projection matrix that will form the basis of a new coordinate system, such iterative process is called sequential projection pursuit (SPP) [67]. To avoid a degenerated solution where all the directions in the projection matrix are equal, orthogonality is imposed as constraint in the optimisation problem of Eq. (2.34) and Eq. (4.1), corresponding to  $\mathbf{A}^T \mathbf{A} = \mathbf{I}$ . This section studies how to impose the orthogonality on multiple projections  $\mathbf{V} = [\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \dots, \boldsymbol{\nu}_b]$  in the kernel-induced feature space by sequentially computing the coefficient matrix  $\boldsymbol{\Gamma} = [\gamma_1, \gamma_2, \dots, \gamma_b]$ .

Different from the structure removal process used in [216], which kernelizes the Gram-Schmidt orthogonalisation procedure in each iteration to obtain the new projection vectors, the discussion in this section starts from the analysis of the variance in the residual subspace at iteration  $j$ . Given the first  $j$  optimal projection vectors  $\{\boldsymbol{\nu}_s^*\}_{s=1}^j$ , let us search for the  $(j+1)$ th projection vector that maximises the variance of the projected residual<sup>1</sup> as follows [49]

$$\boldsymbol{\nu}_{j+1}^* = \operatorname{argmax}_{\|\boldsymbol{\nu}\|=1} \frac{1}{n} \sum_{t=1}^n \left[ \boldsymbol{\nu}^T \left( \boldsymbol{\phi}(\mathbf{x}_t) - \sum_{s=1}^j \boldsymbol{\nu}_s^* \boldsymbol{\nu}_s^{*T} \boldsymbol{\phi}(\mathbf{x}_t) \right) \right]^2, \quad (5.6)$$

which corresponds to searching  $\boldsymbol{\nu}_{j+1}^*$  in the orthogonal complement of the subspace spanned by the  $j$  previously-obtained projection vectors. Equation (5.6) can be rewritten in matrix form as follows:

$$\max_{\|\boldsymbol{\nu}\|=1} \boldsymbol{\nu}^T (\mathbf{I} - \mathbf{V}_j^* \mathbf{V}_j^{*T}) \boldsymbol{\Phi}^T \boldsymbol{\Phi} (\mathbf{I} - \mathbf{V}_j^* \mathbf{V}_j^{*T}) \boldsymbol{\nu}, \quad (5.7)$$

where  $\mathbf{V}_j^* = [\boldsymbol{\nu}_1^*, \boldsymbol{\nu}_2^*, \dots, \boldsymbol{\nu}_j^*]$  is the optimal projection matrix for the first  $j$  projection vectors. Following the definition of  $\mathbf{S}_j^\perp = \mathbf{I} - \mathbf{V}_j^* \mathbf{V}_j^{*T}$  as the residual subspace to simplify notation, the solution to the constrained optimisation problem in Eq. (5.7) can be posed in terms of the Lagrangian

$$F(\boldsymbol{\nu}, \alpha) = \boldsymbol{\nu}^T \mathbf{S}_j^\perp \boldsymbol{\Phi}^T \boldsymbol{\Phi} \mathbf{S}_j^\perp \boldsymbol{\nu} - \alpha (\boldsymbol{\nu}^T \boldsymbol{\nu} - 1), \quad (5.8)$$

where  $\alpha$  is the used Lagrange multiplier. Consequently, the stationary points of Eq. (5.7) need to satisfy

$$\frac{\partial F}{\partial \boldsymbol{\nu}} = \mathbf{S}_j^\perp \boldsymbol{\Phi}^T \boldsymbol{\Phi} \mathbf{S}_j^\perp \boldsymbol{\nu} - \alpha \boldsymbol{\nu} = 0. \quad (5.9)$$

---

<sup>1</sup>To facilitate notation, a centred feature space is assumed. Otherwise, apply Eq. (5.19) to centre the data.

This leads to the eigenvalue problem of the matrix  $\mathbf{S}_j^\perp \Phi^T \Phi \mathbf{S}_j^\perp$ . Incidentally,  $\mathbf{S}_j^\perp$  matches the definition of the orthogonal complement of  $V_j^*$ , this projector maps  $\Phi$  into the subspace orthogonal to  $V_j^*$ , thus by using  $\Phi \mathbf{S}_j^\perp$  to replace the original feature matrix  $\Phi$ , orthogonality can be kept when computing the new  $(j + 1)$ th projection vector in the kernel-induced feature space.

Thus, to impose the orthogonality to the proposed nonlinear evolutionary system, the projected features onto the  $(j + 1)$ th projection vector are computed by modifying Eq. (5.3) as

$$\begin{aligned}
\mathbf{Z}(\gamma_{j+1}) &= \Phi \mathbf{S}_j^\perp \nu_{j+1} \\
&= \Phi \mathbf{S}_j^\perp \Phi^T \gamma_{j+1} \\
&= \Phi (\mathbf{I} - \mathbf{V}_j^* \mathbf{V}_j^{*T}) \Phi^T \gamma_{j+1} \\
&= (\Phi \Phi^T - \Phi \Phi^T \Gamma_j^* \Gamma_j^{*T} \Phi \Phi^T) \gamma_{j+1} \\
&= (\mathbf{K} - \mathbf{K} \Gamma_j^* \Gamma_j^{*T} \mathbf{K}) \gamma_{j+1},
\end{aligned} \tag{5.10}$$

where  $\Gamma_j^* = [\gamma_1^*, \gamma_2^*, \dots, \gamma_j^*]$  is the optimal coefficient matrix already obtained for the first  $j$  projection vectors. Consequently, the determination of the nonlinear residual subspace for the  $(j + 1)$ th projection vector has been successfully reduced to the update of the kernel matrix at the  $(j + 1)$ th iteration:

$$\mathbf{K}^{(j+1)} = \mathbf{K} - \mathbf{K} \Gamma_j^* \Gamma_j^{*T} \mathbf{K}. \tag{5.11}$$

Then, the optimisation problem in Eq. (5.4) for the  $(j + 1)$ th projection vector can be written as

$$\gamma_{j+1}^* = \underset{\gamma \in \mathbb{R}^n}{\operatorname{argmax}} \mathfrak{S}(\mathbf{K}^{(j+1)} \gamma). \tag{5.12}$$

This process is repeated until the  $\gamma_{j+1}^*$  does not contribute any more to explain the underlying data structure. Such assessment is carried by a criterion which relies on Bayesian model selection [21], and it is based on the fact that the remaining structure on the residual subspace is decreased as the number of projections increases. The Bayesian stopping criterion (BSC) is defined as

$$\beta = \left( 2^{n \times [\mathfrak{S}(\mathbf{K}^{(j)} \cdot \gamma_j^*) - \mathfrak{S}(\mathbf{K}^{(j+1)} \cdot \gamma_{j+1}^*)]} + 1 \right)^{-1}. \tag{5.13}$$

The  $(j+1)$ th projection is included if  $\beta$  is bigger than a predefined threshold  $\delta$ , otherwise SPP stops with  $j$  obtained projections.

## 5.4 Whitening in Feature Space

As it is known, the whitening process is required to decorrelate the data previous to the sequential induction process [60]. To whiten the data in the non-observable feature space, let  $\mathbf{C}_\Phi$  denote the covariance matrix for the training samples calculated in the

kernel space  $\mathcal{H}$ . By applying the standard whitening technique, the whitening matrix in the non-observable feature space is

$$\mathbf{W} = \mathbf{L}^{-1/2}\mathbf{M}^T \quad (5.14)$$

where  $\mathbf{L}$  is a diagonal matrix with the  $j^{\text{th}}$  largest eigenvalue of  $\mathbf{C}_\Phi$  as its  $j^{\text{th}}$  diagonal element, and  $\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_j, \dots]$  with the  $j^{\text{th}}$  eigenvector of  $\mathbf{C}_\Phi$  corresponding to the  $j^{\text{th}}$  largest eigenvalue as its  $j^{\text{th}}$  column. Since  $\mathbf{C}_\Phi$  is calculated in  $\mathcal{H}$  and thus possesses an infinite size, it is not straightforward to directly obtain  $\mathbf{L}$  and  $\mathbf{M}$ . Schölkopf et al [51] show that both  $\mathbf{L}$  and  $\mathbf{M}$  can be approximated by computing the eigen-decomposition of the kernel matrix  $\mathbf{K}$  between the training samples, given as

$$\mathbf{L} = \frac{1}{n}\mathbf{\Lambda}, \quad (5.15)$$

$$\mathbf{M} = \Phi^T \Psi \quad (5.16)$$

where  $\mathbf{\Lambda}$  is a diagonal matrix with the  $j^{\text{th}}$  largest eigenvalue of  $\mathbf{K}$  as its  $j^{\text{th}}$  diagonal element, and  $\Psi = [\psi_1, \psi_2, \dots, \psi_j, \dots]$  with the  $j^{\text{th}}$  eigenvector of  $\mathbf{K}$  corresponding to the  $j^{\text{th}}$  largest eigenvalue as its  $j^{\text{th}}$  column. Since the orthogonality condition between the eigenvectors of the covariance matrix is required, matrix  $\mathbf{M}$  is further scaled by  $\mathbf{\Lambda}^{\frac{1}{2}}$ , as

$$\mathbf{M} = \Phi^T \Psi \mathbf{\Lambda}^{\frac{1}{2}}, \quad (5.17)$$

so that  $\mathbf{M}^T \mathbf{M} = \mathbf{I}$ . By incorporating both Eq. (5.14) and Eq. (5.17), the dot product of the whitened data in feature space can be expressed as

$$\hat{\mathbf{K}} = \Phi \mathbf{W}^T \mathbf{W} \Phi = n \Psi \Psi^T. \quad (5.18)$$

To facilitate the notation, the learning data was assumed to be centered in the feature space. When such assumption is not valid, the previous results still hold but the kernel matrix  $\mathbf{K}$  need to be replaced by

$$\bar{\mathbf{K}} = \mathbf{K} - \mathbf{1}_n \mathbf{K} - \mathbf{K} \mathbf{1}_n + \mathbf{1}_n \mathbf{K} \mathbf{1}_n \quad (5.19)$$

where  $\mathbf{1}_n$  is a  $n \times n$  matrix with all its elements equal to  $1/n$ .

In the following, two modalities of the proposed extension of PP to a kernel-induced feature space will be distinguished. The first modality will be referred as kernel projection pursuit (KPP), where the projection index  $\mathfrak{S}$  in Eq. (5.4) can be any of the existing indices introduced in Section 2.3.3.2. The second modality will be referred as evolutionary kernel projection pursuit (EKPP), where the evolutionary framework described in Section 4.2 is used to infer an optimal projection index. This distinction will help us to highlight the advantages of an evolved PP index against its non-optimal counterpart.

## 5.5 Experimental Results

### 5.5.1 Datasets

Six datasets were used to benchmark the proposed algorithm for nonlinear feature extraction, including arcene, dexter, dorothea, madelon, duke and PIE. The first four datasets were from the NIPS'03 challenge [219], of which each is provided as three separate splits, one for training, one for validation and one for testing. However, only the labels corresponding to the training and validation sets are publicly available, the labels for the testing split have been retained by the challenge's organisers to encourage post-challenge submissions. So the provided training and validation sets were merged into a single dataset for experiments. Among the rest two datasets, duke was taken from the work presented in [220] and PIE is a modified subset of the CMU PIE database as used in [13].

In order to provide a fair comparison between different feature extraction methods, a 10-CV partition scheme for final model assessment was implemented. Specifically, each dataset  $\mathbf{D}$  is divided into two mutually exclusive sets  $\mathbf{D} = \Upsilon \cup \tilde{\Upsilon}$ . The learning set  $\Upsilon$  is used within the induction process to derive an optimal model, performing model selection via the  $h$ -CV (further splitting  $\Upsilon$  into training and validation sets) and the genetic search. Once the optimal model is known, the testing set  $\tilde{\Upsilon}$  is processed with the fine tuned feature extraction algorithm and classified using the optimal classifier parameters, then its balanced error rate (BER) is accumulated. This process is repeated over each fold of the 10-CV for final model assessment. In Table 5.1, the size of the testing, training and validation partitions can be observed for each dataset as used in the present work, as well as the number of features in the original feature set, the total number of classes and the ratio of the number of in-class samples to the number of out-class samples, named as class imbalanced ratio. A low ratio value represents a highly imbalanced dataset.

To investigate the potential separability of the used datasets in advance and assess their suitability for nonlinear feature extraction, a preliminary study was performed using the popular classifier SVM in two modalities. The first modality consisted in a linear SVM ( $l$ -SVM), where no feature extraction was applied to the data. The second modality was the SVM with Gaussian kernel ( $g$ -SVM) without feature extraction, that performs nonlinear classification by translating the maximum-margin hyperplane to a transformed feature space by means of the kernel trick. Since previous research has shown the effectiveness of feature selection algorithms for classifying the NIPS'03 datasets, feature selection by sequential forward selection was carried using mutual information as cost function followed by  $g$ -SVM (MI+ $g$ -SVM) as a comparison point. The BER obtained by these three classifiers are shown in Table 5.1, from which it can be observed (1) the nonlinear classifier outperforms its linear version, which indicates the

Table 5.1: Testing, training and validation set sizes, number of features, total number of classes and class imbalanced ratio, as well as the balanced error rate for each problem.

Dataset	Data Information						Balanced Error Rate		
	No. Test	No. Train	No. Validation	No. Feature	No. Class	Imbalanced Ratio	I-SVM	g-SVM	MI+g-SVM
arcene	20	120	60	10,000	2	0.78	27.94	23.94	12.26
dexter	60	360	180	20,000	2	1.00	18.46	13.46	17.70
dorothea	115	690	345	100,000	2	0.11	37.23	23.12	18.86
madelon	260	1560	780	500	2	1.00	41.10	24.87	51.23
duke	4	26	14	7129	2	1.09	12.26	8.33	19.0
PIE	1155	6933	3466	1024	68	0.12	51.03	43.12	46.37

classification task could benefit from nonlinear feature extraction, (2) the classification task does not always benefit from feature selection, specially when features apparently meaningless by their own are discarded but when combined with others boost the classification performance.

### 5.5.2 Experimental Setup

This section elaborates on the comparison between EKPP and three existing nonlinear feature extraction techniques that also employ the kernel trick to achieve nonlinearity, namely KLPP [74], KFD [92] and KPCA [51], as well as KPP with predetermined index, and two linear feature extraction methods. The first linear extraction method is EPP [221] and the other is PCA. The comparison was based on the model assessment scheme described in Section 5.5.1. An LDA classifier was used to compute the class labels of the testing samples. It assumes equal prior probabilities for all classes and fits a multivariate normal density to each group, with a pooled estimate of the covariance. Different from the kNN classifier used in the previous chapter, the use of an LDA classifier obeys the fact that kNN may diminish the effect of the extracted features by building nonlinear class-boundaries. Therefore, to favour the effects of the feature extraction stage, LDA was selected as classification algorithm, which generates linear decision boundaries. The kernel function used by the nonlinear extraction methods follows the recommendations in [222], and was designed to be a mix between the gaussian and the polynomial kernel, defined as

$$k_{ij} = \left(\theta_1 + \mathbf{x}_i \mathbf{x}_j^T\right)^{\theta_2} \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / \theta_3\right), \quad (5.20)$$

where  $\theta_1, \theta_2$ , and  $\theta_3$  allow the selection of the optimal type of kernel to be used in each method.

The scalar part of the hybrid chromosome in EKPP and the chromosome used to fine-tune the hyper-parameters in the competing methods share the same structure. Such structure was designed to represent the parameters of each model (i.e., coefficients  $\{\theta_i\}_{i=1}^3$  of the kernel function, and a scalar  $b$  representing the number of retained eigenvectors  $\psi_j$  from the kernel matrix) as binary words. The widths for each variable  $\{\theta_i\}$  are 4 bits, 4 bits, and 8 bits respectively, while the width for  $b$  depends on the number of training samples, and was determined according to the formula  $\lceil \log_2(2\|\text{Ind}(\Omega_i)\|/3) \rceil$ , where  $\|\text{Ind}(\Omega_i)\|$  is the number of samples in the training set, and  $\lceil \cdot \rceil$  rounds its argument to the nearest upper integer.

EKPP was implemented with the help of GPLAB [210] library for MATLAB. The GP algorithm was run over 100 generations with a population of 20 individuals, with a fix cross-over and mutation rate of 90% and 10% respectively. The number of folds defining Eq. (5.5) was fixed to  $h = 3$  for all experiments with EKPP, and the threshold  $\delta$  used in the stopping criterion was set to 0.35, as suggested in [21]. The number  $b$  of eigenvectors  $\psi_j$  from the kernel matrix used to compute  $\hat{\mathbf{K}}$  was restricted to be less

than the rank of  $\mathbf{K}$  to avoid singularity problems, thus the final number of projections, although determined by the BSC in the scheme described in Section 5.3, can not be more than  $b$ .

For KPP, KLPP, KFD and KPCA, parameter fine-tuning was implemented by means of a GA, which was favoured over typical grid search based on the results of preliminary trials, that reported a considerable saving in computational time. Additional experiments were carried to determine the optimal crossover and mutation rate such that the local optima problem was avoided. The best performing combination, out of four empirical setups, resulted in probabilities of 10% and 90% for mutation and crossover. In KPCA  $b$  is the final number of projections; the genetic search implemented for KFD ignore the variable  $b$  since this method computes at most  $(c - 1)$  number of projection components; while for KLPP, an extra variable was added to encode the number of neighbours used to compute the affinity matrix. To compare the advantages of an optimal evolved index over its non-optimal counterpart,  $L_r$ -norm index was used with  $r = 2$  as projection index in the scheme described in Section 5.3, this setup was labeled as KPP. The same restrictions on  $b$  as in EKPP are followed here to avoid singularity problems.

Considering the selection of the classification algorithm could impact the performance of the proposed method, experiments using EKPP followed by the kNN classifier were also performed, referred as EKPP<sub>k</sub>. The changes are mainly reflected on the structure of the scalar part in the hybrid chromosome, including the hyper-parameters of the classifier (i.e., the coefficient for the Minkowsky distance, and the number of nearest neighbours), and the width for a gaussian kernel. The classifier hyper-parameters were encoded as 4-bit binary words, while an 8-bit binary word was reserved for the width of the gaussian kernel. The number of eigenvectors  $\psi_j$  used to compute  $\hat{\mathbf{K}}$  was fixed to ten, and then SPP was used on these feature space to extract relevant features.

### 5.5.3 Experimental Results and Analysis

The median and IQR of the balanced error rate over a 10-CV are provided in Table 5.2. Such statistics were obtained from the classification tasks of the six datasets, using the features extracted with EKPP, KPP, KPCA, KFD, KLPP, EPP and PCA in combination with an LDA classifier. The number of extracted features and relevant statistics regarding computational time can be observed in Table 5.3.

Performance of a feature extraction algorithm is not only characterised by the classification error they present, but also by the compression rate achieved. Similarly, the information embedded in the extracted features is not always optimal for the classification task at hand, thus facing an intrinsic multi-objective optimisation problem in the design, training and selection of such algorithms. By considering both the classification error and compression ability of a feature extraction algorithm, a fair comparison is

Table 5.2: Comparison of the median and IQR of the balanced error rate over a 10-CV for different feature extraction algorithms. The best performance is marked in bold.

Method	arcene		dexter		dorothea		madelon		duke		PIE	
	Median	IQR	Median	IQR	Median	IQR	Median	IQR	Median	IQR	Median	IQR
EPP	16.67	12.49	36.57	3.47	18.27	7.10	36.57	3.47	7.62	4.72	36.44	2.72
PCA	20.37	18.66	22.98	5.62	41.19	4.63	22.98	5.62	7.01	3.21	40.54	8.23
KPP	24.49	16.67	17.69	9.69	24.21	10.32	24.32	11.10	6.12	4.11	34.78	6.76
KPCA	40.83	28.28	33.15	11.25	50.00	7.14	31.21	6.56	8.21	6.10	35.82	7.14
KFD	16.38	19.49	18.86	22.48	20.08	5.37	24.74	1.12	<b>4.87</b>	3.83	27.85	12.47
KLPP	20.44	6.52	19.94	7.08	17.10	11.36	24.83	3.11	9.63	6.98	53.81	10.26
EKPP	<b>10.10</b>	4.32	<b>11.40</b>	5.83	<b>11.92</b>	8.36	<b>23.20</b>	4.92	5.43	2.71	<b>27.17</b>	7.06
EKPP <sub>k</sub>	12.10	5.00	13.33	5.00	23.74	18.85	38.13	7.50	6.59	3.02	47.40	8.00



Table 5.3: The number of extracted features, the time needed to perform model selection Time1 (in hrs.), and the time needed to extract features with the optimal model Time2 (in sec.) for different algorithms.

arcene				dexter				dorothea			
	No. Feature	Time1	Time2		No. Feature	Time1	Time2		No. Feature	Time1	Time2
EPP	20.28	0.30	16.00	EPP	2.00	1.21	5.23	EPP	2.00	3.88	1.27
PCA	56.00	0.43	2.16	PCA	6.00	1.75	3.02	PCA	503.08	10.12	33.63
KPP	3.00	3.15	10.00	KPP	3.00	1.96	7.14	KPP	3.20	5.79	19.37
KPCA	14.70	6.11	23.00	KPCA	7.20	4.81	11.47	KPCA	447.00	7.54	31.17
KFD	1.00	1.72	10.00	KFD	1.00	2.60	5.62	KFD	1.00	4.63	1.67
KLPP	116.14	6.17	22.20	KLPP	86.03	3.14	7.21	KLPP	329.00	8.12	23.33
EKPP	3.00	6.81	19.00	EKPP	3.00	3.22	6.27	EKPP	3.40	8.21	16.08
EKPP <sub>k</sub>	9.70	5.61	20.00	EKPP <sub>k</sub>	10.00	3.98	9.19	EKPP <sub>k</sub>	2.90	8.43	21.75

madelon				duke				PIE			
	No. Feature	Time1	Time2		No. Feature	Time1	Time2		No. Feature	Time1	Time2
EPP	163.09	8.35	19.14	EPP	3.09	3.04	4.01	EPP	211.09	17.49	31.49
PCA	278.00	5.16	18.59	PCA	4.02	4.20	4.14	PCA	250.00	19.83	42.55
KPP	6.02	4.03	5.93	KPP	3.10	1.81	1.21	KPP	63.09	7.33	9.61
KPCA	7.11	8.97	9.12	KPCA	20.09	2.15	11.6	KPCA	332.23	6.31	9.17
KFD	1.00	4.15	1.23	KFD	1.00	2.23	1.07	KFD	67.00	7.06	9.12
KLPP	123.70	8.42	11.17	KLPP	26.00	4.63	5.28	KLPP	265.02	5.23	8.86
EKPP	2.60	10.13	1.43	EKPP	7.60	2.33	5.23	EKPP	12.43	11.40	20.53
EKPP <sub>k</sub>	1.30	4.81	1.37	EKPP <sub>k</sub>	10.00	2.69	6.07	EKPP <sub>k</sub>	10.00	6.84	10.14

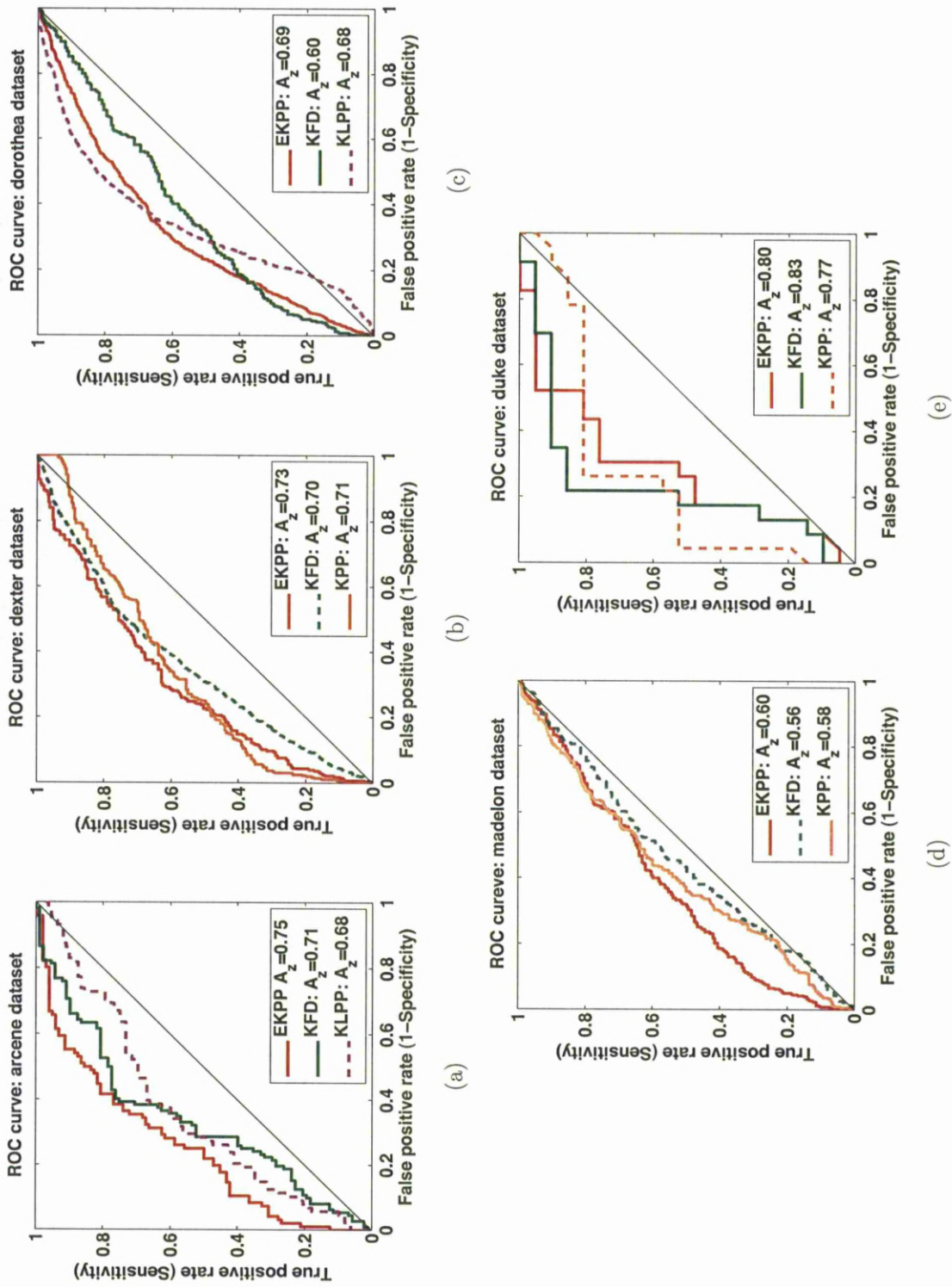


Figure 5.1: Examples of ROC curves for the three best performing methods for (a) arcene, (b) dexter, (c) dorothea, and (d) madelon and (e) duke. EKPP is plotted in continuous red line and the method ranked in third place is in dash line.

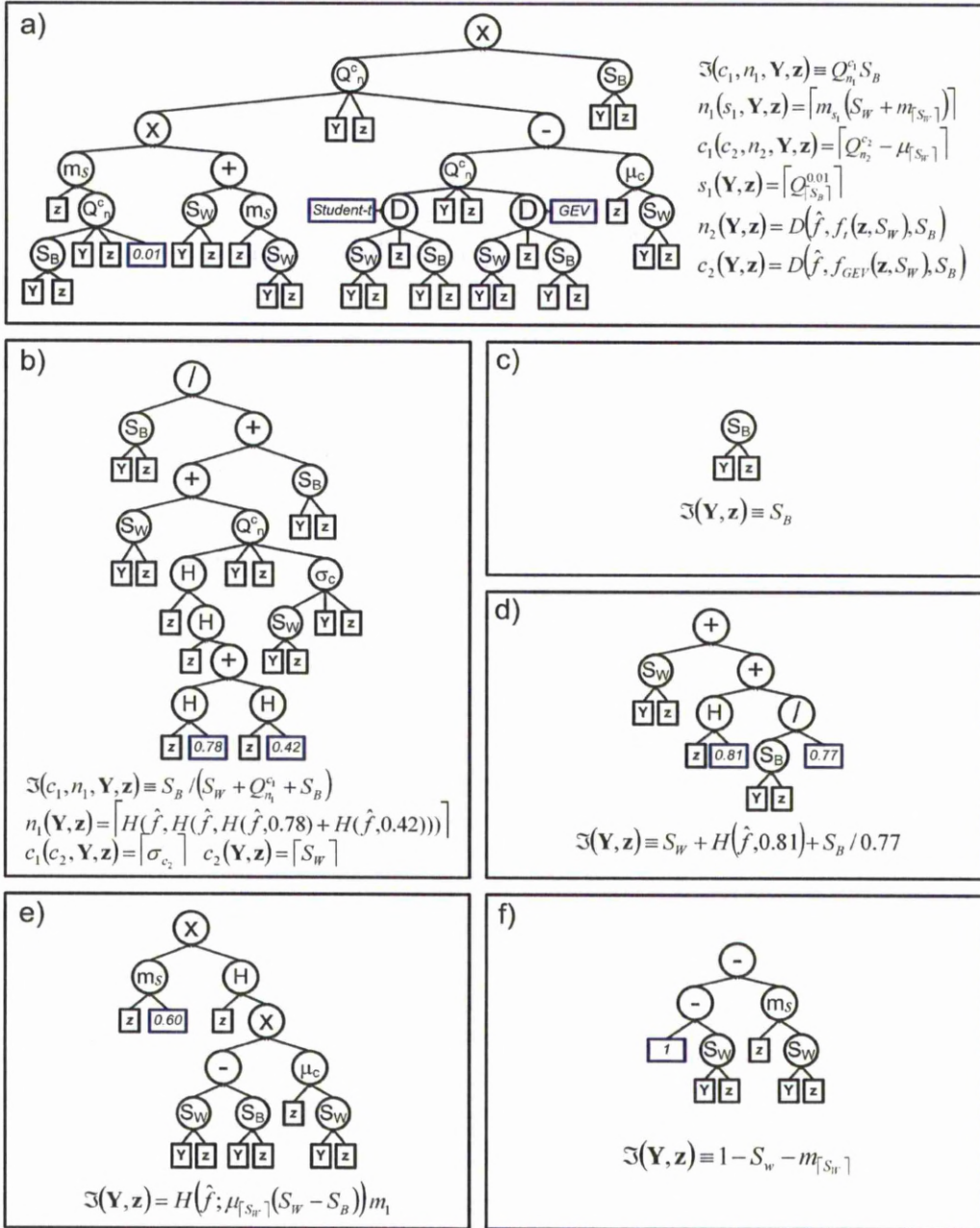


Figure 5.2: Evolved trees for different datasets. (a) arcene, (b) madelon, (c) dexter, (d) dorothea, (e) PIE, and (f) duke datasets

achieved where all the algorithms have been fine-tuned by an evolutionary framework, thus comparing them at their Pareto optimal point.

As can be seen in Table 5.2, EKPP achieves the lowest classification error for most of the datasets as compared to not only the four nonlinear and two linear feature extraction methods but also the three linear/nonlinear SVM classifiers with/without feature selection (see Table 5.1). In general, it is expected that nonlinear extraction algorithms would perform better than linear ones. This fact is corroborated for the proposed evolutionary method in Table 5.2, where EPP shows a lower classification performance than EKPP.

Considering both the classification accuracy shown in Table 5.2 and the compression rate indicated by the number of extracted features in Table 5.3, on average EKPP outperforms most of the algorithms in all datasets, being close-followed by KFD and much better than KPP, KPCA and KLPP. Although the compression rate of EKPP is not always the best, as compared with KFD, the information embedded in the extracted features helped to improve the classification performance.

The performance of KLPP among the kernel methods is not as good as expected. In the existing literature, LPP has been reported to have an outstanding performance for face recognition problems [13], nevertheless since LPP effectively unfolds the nonlinear structure of the manifold, applying a second nonlinear mapping by means of the kernel trick may degrade the discriminatory information. A similar deterioration can be observed in the classification performance of  $EKPP_k$ . Although in this case there are several possible reasons that could have caused such low performance, the burden in selecting a classifier with higher number of parameters does not show a considerable improvement over the simpler EKPP setup.

In order to provide a sense of the complexity of EKPP compared to the competing methods, relevant computational times are provided in Table 5.3. Time1 is the time taken for the evolutionary search to evolve an optimal model in one of the folds of the 10-CV. Time2 is the time taken by the optimal model to perform feature extraction and classification of the testing set. EKPP shows better compression rate and classification accuracy in Table 5.2. Although this improvement is at expense of an increase in the induction and classification time, the overall computation time (Time1+Time2) is comparable to that of the competing feature extraction algorithms fine-tuned by a GA.

Additionally to the classification accuracy analysis, a sensitivity analysis was also performed over the extracted features for the three best feature extraction methods for the binary classification problems. Every extracted feature was ranked using its class scatter ratio [42], and the receiver operating characteristic (ROC) curve was computed for the dominant feature. Figure 5.1 display such curves for each dataset along with the area under ROC curves ( $A_z$ ). The features found by EKPP work much better together than individually (see Table 5.2 and Figure 5.1), nevertheless the best ranked feature

managed to perform better than the competing methods (see Figure 5.1).

The inferred PP indices are showed as a tree representation along with their corresponding equations in Fig. 5.2. All the presented trees are highly supervised, with the tree corresponding to dexter dataset being only the definition of the between-class scatter. It can be seen that trees for highly imbalanced datasets strongly depend on Renyi entropy, while those trees for balanced datasets depend on functions of the within-class and between-class scatters.

## 5.6 Summary

In this chapter we have discussed a nonlinear extension of the proposed evolutionary framework by means of a kernel-induced feature space. Its main advantage, compared to the method proposed in [216], is the determination of the nonlinear residual subspace in SPP via a simple matrix updating formula (given in Section 5.3) which speeds up computation of the projection matrix. Additionally, whitening in feature space was expressed in terms of the eigenvectors corresponding to the largest eigenvalues of the kernel matrix (Section 5.4). It was experimentally shown that EKPP outperforms existing kernel methods, as well as its linear version EPP, for most of the selected high-dimensional classification problems (see Table 5.2).

Besides inducing an optimal projection index, the proposed evolutionary framework also optimises the type of kernel function suited to the application at hand. The selection of the kernel function is controlled by the parameters of Eq. (5.20), which effectively combine two popular kernel functions, namely Gaussian and polynomial kernel. Four other kernel methods were compared against EKPP (i.e. KPP, KPCA, KFD, and KLPP), which parameters were also optimised with an evolutionary algorithm to provide a fair comparison. Among them, KLPP exhibited the worst compression rate, while EKPP showed high compression rates, ranging from 98.79% to 99.99%, without sacrificing accuracy.

Selection of the classifier plays an important role in the performance of the evolved classification system. Two classifiers were tested as part of the fitness function showed in Eq. (5.5), an LDA and a kNN. Classification systems evolved using an LDA outperformed those evolved with a kNN, this could be attributed to the fact that kNN builds nonlinear classification boundaries that diminish the effect of the kernel-induced nonlinear extracted features. Thus, in order to take the very best out of EKPP, it is recommended to only use linear classifiers when inducing classifier systems via the proposed evolutionary framework. As with its predecessor, EKPP suffers from large induction times, thus it can not be used for online applications.

## Chapter 6

# Evolutionary Induction of Heterogeneous Proximities for Supervised Embeddings

The discussion so far has been centred on a generalization of linear projection techniques. An evolutionary framework has been described, able to automatically design a projection index to be used as feature extraction stage for a given classification problem. Additionally, the original projection pursuit problem has been mapped to a kernel-induced feature space with help of Mercer's theorem, unfolding any nonlinearities and improving classification accuracy for nonlinear datasets. As result of using a kernel-induced feature space, the small sample size problem for high-dimensional datasets is avoided. However, recent evidence suggest that non-observable spaces allow nonlinear embeddings of data originally laying in a low-dimensional manifold [68]. When this assumption holds, a linear projection, or nonlinear projection via kernel-induced spaces, fails to unveil the true low-dimensionality of the embedded manifold. Spectral embedding methods, described in Section 2.3.4, have been successfully used to extract features able to describe nonlinear manifolds embedded into non-observable spaces. Their design is based on correctly modelling the local geometry for each sample, which will be retained in the resulting low-dimensional space. The present chapter seeks to address the problems related to the automatic design of supervised spectral embedding (SSE) methods. First an alternative formulation to the classic trace optimisation problem, used by spectral embedding methods, is proposed. Then, the design process is modelled as a complex learning task, where an optimal model is induced from a set of training data. The inducer engine was implemented using an evolutionary search, capable of building not only a family of existing spectral embedding methods, but also create new models as a result of its powerful syntax and high expressive lexicon. The performance of several human-engineered SSE methods is compared with these automatically generated embeddings for the task of classification, and results show a decrease in the classification error.

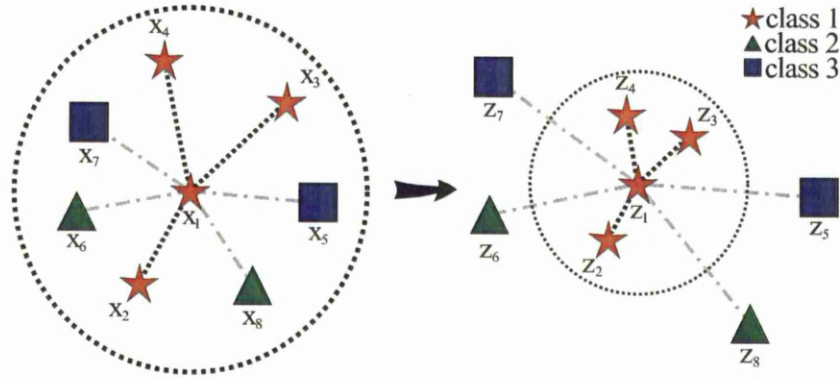


Figure 6.1: An example of a possible SSE configuration of the original space (left) to the embedding one (right), where the three friends of  $\mathbf{x}_1$  are pulled closer, while its four enemies are pushed afar.

## 6.1 Model Definition

If we assume a set  $\{\mathbf{x}_i \in \mathbb{R}^m\}_{i=1}^n$  of training samples corresponding to discrete labels  $y_i \in \{1, \dots, c\}$ , an SSE algorithm generates  $n$  embeddings  $\{\mathbf{z}_i \in \mathbb{R}^b\}_{i=1}^n$  of  $b \ll m$  dimensions each. These two sets can also be conveniently denoted through the original  $n \times m$  feature matrix  $\mathbf{X} = [x_{ij}]$  and the  $n \times b$  embedding matrix  $\mathbf{Z} = [z_{ij}]$ , with row vectors the original samples  $\mathbf{x}_i$  and the embeddings  $\mathbf{z}_i$ , respectively.

The supervised character of an SSE method implies that its objective is the creation of a new configuration in the embedding space, where the class structures and separabilities existing in the original space are not only maintained, but also reinforced. As it is demonstrated in Fig. 6.1, this is achievable by increasing the similarities between friends (samples from the same class), while making enemies (samples from different classes) more distant. A straightforward way of implementing this behaviour, is to minimise the weighted sum of all the pairwise embedding distances, as in the standard unsupervised embedding, according to

$$\min_{\{\mathbf{z}_i \in \mathbb{R}^b\}_{i=1}^n} \frac{1}{2} \sum_{i,j=1}^n w_{ij} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2, \quad (6.1)$$

or equivalently

$$\min_{\mathbf{Z} \in \mathbb{R}^{n \times b}} \text{trace} [\mathbf{Z}^T \mathbf{L} \mathbf{Z}], \quad (6.2)$$

but with the similarity weights  $w_{ij}$  corresponding to enemy pairs  $(i, j)$  treated differently than the friend ones. In Eq. (6.2),  $\mathbf{W} = [w_{ij}]$  is the overall similarity matrix, and  $\mathbf{L} = \mathbf{D}(\mathbf{W}) - \mathbf{W}$  is the standard Laplacian matrix, where  $\mathbf{D}(\mathbf{W})$  gives a diagonal matrix with elements  $d_{ii} = \sum_{j=1}^n w_{ij}$ .

Below, a few examples are given on how different SSE methods use  $\mathbf{W}$  to directly control and differentiate the friend and enemy vicinities. For comparison purposes,

all methods are re-express in terms of their weights  $w_{ij}$ . The first of these methods, the discriminant neighbourhood embedding (DNE) [83] is based on a sample weighting defined as

$$w_{ij} = \begin{cases} +1 & \text{if } \mathbf{x}_j \in N_F(\mathbf{x}_i, k) \vee \mathbf{x}_i \in N_F(\mathbf{x}_j, k), \\ -1 & \text{if } \mathbf{x}_j \in N_E(\mathbf{x}_i, k) \vee \mathbf{x}_i \in N_E(\mathbf{x}_j, k), \\ 0 & \text{otherwise,} \end{cases} \quad (6.3)$$

where  $N_F(\mathbf{x}_i, k)$  represents the  $k$ -nearest friends of  $\mathbf{x}_i$ , and  $N_E(\mathbf{x}_i, k)$  its  $k$ -nearest enemies. The supervised optimal LPP (SOLPP) method [223], integrates class prior probabilities into the weighting matrix as

$$w_{ij} = \begin{cases} p_{y_i}^2 s_{ij}(1 + s_{ij}) & \text{if } \mathbf{x}_j \in N_F(\mathbf{x}_i, k) \vee \mathbf{x}_i \in N_F(\mathbf{x}_j, k) \\ p_{y_i} p_{y_i} s_{ij}(1 - s_{ij}) & \text{if } \mathbf{x}_j \in N_E(\mathbf{x}_i, k) \vee \mathbf{x}_i \in N_E(\mathbf{x}_j, k), \\ 0 & \text{otherwise,} \end{cases} \quad (6.4)$$

where  $s_{ij} = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\tau}\right)$  is a Gaussian-based similarity controlled by the parameter  $\tau$ , and  $p_k$  ( $k = 1, 2, \dots, c$ ) is the  $k^{\text{th}}$  class prior probability. Another method is the repulsion OLPP (OLPP-R) [78], based on a Laplacian  $\mathbf{L} = \mathbf{L}_c - \beta \mathbf{L}_r$ , defined as the linear combination of a class  $\mathbf{L}_c$  and a repulsion Laplacian  $\mathbf{L}_r$ , for some user-defined parameter  $\beta > 0$ . This is equivalent to the calculation of the weights according to

$$w_{ij} = \begin{cases} \frac{1}{n_l} & \text{if } y_i = y_j = l, \\ -\beta s_{ij} & \text{if } [\mathbf{x}_j \in N(\mathbf{x}_i, k) \vee \mathbf{x}_i \in N(\mathbf{x}_j, k)] \wedge y_i \neq y_j, \\ 0 & \text{otherwise,} \end{cases} \quad (6.5)$$

where the alternative weight  $s_{ij} = \left(\tau + \frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\|\mathbf{x}_i\|_2^2 + \|\mathbf{x}_j\|_2^2}\right)^{-1}$  is used to define the repulsion Laplacian,  $n_l$  is the number of samples in class  $l$ , and  $N(\mathbf{x}_i, k)$  represents the  $k$ -nearest neighbors of  $\mathbf{x}_i$ . Supervised ONPP (SONPP) [72] is a method that attempts to reconstruct each sample by a linear combination of its friends, thus, only takes into account similarities between friends based on

$$w_{ij} = \begin{cases} \tilde{w}_{ij} & \text{if } y_i = y_j, \\ 0 & \text{otherwise,} \end{cases} \quad (6.6)$$

where  $\tilde{w}_{ij} = m_{ij} + m_{ji} - \sum_{k=1}^n m_{ki} m_{kj}$  is computed from the reconstruction coefficient matrix  $\mathbf{M} = [m_{ij}]$ , and  $\mathbf{M}$  is obtained by minimising the reconstruction error as follows

$$\begin{aligned} \min_{\mathbf{M} \in \mathbb{R}^{n \times n}} \quad & \sum_{i=1}^n \left\| \mathbf{x}_i - \sum_{j=1}^n m_{ij} \mathbf{x}_j \right\|_2^2, \\ \text{s.t.} \quad & m_{ij} = 0 \text{ if } \mathbf{y}_i \neq \mathbf{y}_j, \sum_{j=1}^n m_{ij} = 1. \end{aligned} \quad (6.7)$$

The matrix  $\mathbf{M}$  will have a block diagonal form  $\mathbf{M} = \text{diag}(\mathbf{M}_1, \mathbf{M}_2, \dots, \mathbf{M}_c)$  after a simultaneous re-ordering of the rows and columns to keep intra-class samples together, where the  $l^{\text{th}}$  block corresponds to the  $l^{\text{th}}$  class. The repulsion ONPP (ONPP-R) [78]



amplifies neighboring enemy dissimilarities by incorporating the repulsion Laplacian into SONPP using

$$w_{ij} = \begin{cases} \tilde{w}_{ij} & \text{if } y_i = y_j, \\ -\beta s_{ij} & \text{if } [\mathbf{x}_j \in N(\mathbf{x}_i, k) \vee \mathbf{x}_i \in N(\mathbf{x}_j, k)] \wedge y_i \neq y_j, \\ 0 & \text{otherwise,} \end{cases} \quad (6.8)$$

where  $s_{ij}$  is the alternative weight as used by OLPP-R. Finally, the discriminant ONPP (DONPP) [80] is very similar to ONPP-R, but defines the neighboring enemies and their corresponding weights differently as

$$w_{ij} = \begin{cases} \tilde{w}_{ij} & \text{if } y_i = y_j, \\ -\beta & \text{if } \mathbf{x}_j \in N_E(\mathbf{x}_i, k) \vee (\text{or } \wedge) \mathbf{x}_i \in N_E(\mathbf{x}_j, k), \\ 0 & \text{otherwise.} \end{cases} \quad (6.9)$$

Although the aforementioned SSE methods are based on different ways of controlling the friend and enemy vicinities, they are all relying of a uniform treatment of the weights between friends, and between enemies. In this way, all friends regardless of the class they belong to, are assigned weights calculated via a common metric. Similarly, the weights for the enemy pairs are homogeneously calculated, regardless of the class pairs containing them. The underlying assumption is that samples from each class lie on manifolds having similar geometrical configurations and densities. However, real world datasets may contain multiple, intersecting or partially overlapped manifolds with different orientations and densities [224, 225]. Additionally, the inter-class configurations may also vary, so that the enemy dissimilarities cannot be treated in a homogeneous manner for all class pairs. Fig. 6.2 exemplifies such a possible scenario.

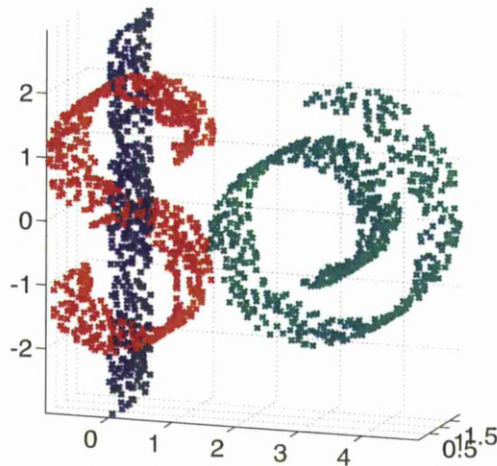


Figure 6.2: Synthetic 3D dataset (based on [224]) composed of three classes (the '3' and '9' parts in the dollar shape and the Swiss roll) with samples lying on different manifolds.

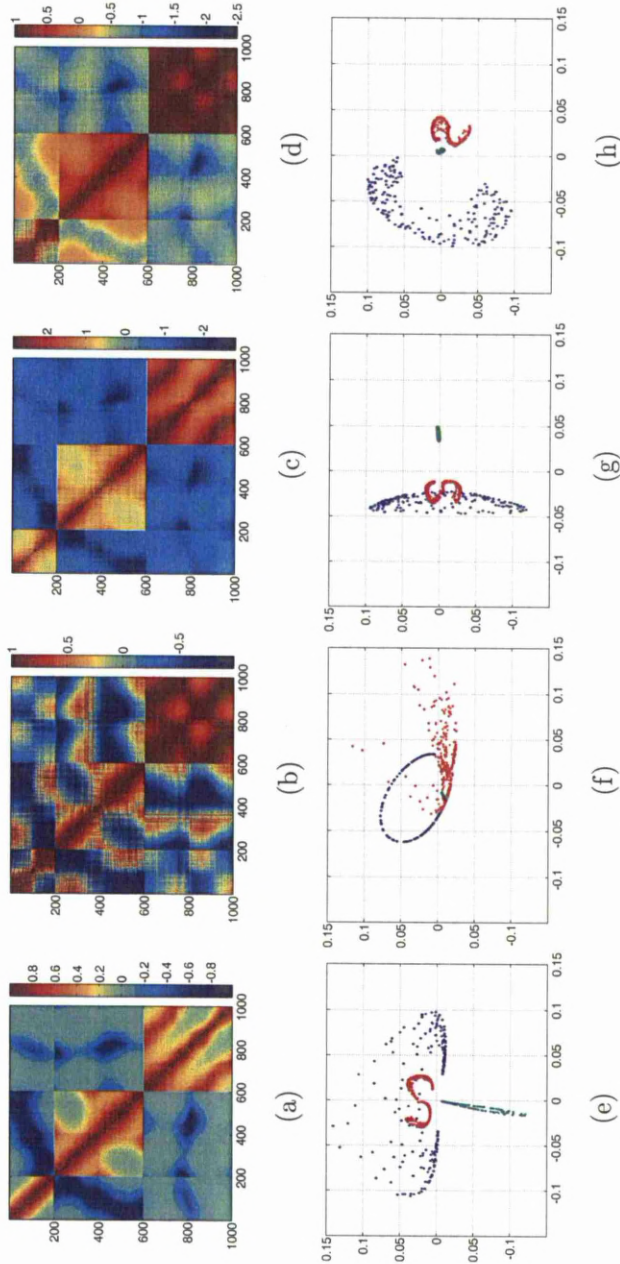


Figure 6.3: Proximity matrices  $\mathbf{W}$  in (a)-(d) and their corresponding 2D embeddings in (e)-(h), from processing the dataset of Fig. 6.2. A fully connected graph is used to calculate the  $w_{ij}$  weights using: (a) the Gaussian kernel  $s_{ij}$  ( $\tau = 1.5$ ), (b) the cosine similarity, (c) the alternative weight [78] ( $\sigma = 0.35$ ), and (d) a  $\mathbf{W}$  composed of these three. All matrices are re-ordered according to the three class labels for display purposes, and are partitioned symmetrically by  $3 \times 3$  blocks. Friend pairs correspond to the diagonal blocks, while enemy ones to the off-diagonal blocks. The matrix  $\mathbf{W}$  of (d) is assembled from blocks taken from the matrices of (a)-(c) as follows:  $\mathbf{W}_{12}$ ,  $\mathbf{W}_{22}$  are Gaussian similarities from (a),  $\mathbf{W}_{11}$ ,  $\mathbf{W}_{33}$  are cosine from (b), and  $\mathbf{W}_{13}$ ,  $\mathbf{W}_{23}$  alternative weights from (c). Embedding generation was based on a supervised version of LPP with all edges connecting enemies assigned negative similarities.

In such cases, an SSE algorithm based on standard weight calculations, may display poor class separability and fail to generate embeddings that appropriately maintain or reinforce friend proximity and enemy remoteness. Fig. 6.3 provides a representative demonstration of this situation for the dataset of Fig. 6.2. Figs. 6.3(a-c) display the similarity matrices  $\mathbf{W}$  using three popular homogeneous proximity measures. Using negatively weighted enemy similarities to obtain dissimilarities, the three corresponding two-dimensional embeddings in Figs. 6.3(e-g) were generated. It can be seen that the separability in the embedding spaces is not enforced for all three classes. In Fig. 6.3(d) the similarity matrix  $\mathbf{W}$  is a heterogeneous block composition of the other three matrices, where the different friend (diagonal) and enemy (off-diagonal) blocks are calculated using different proximity measures. In the corresponding embeddings of Fig. 6.3(h), it can be seen that all three classes are sufficiently separated, leading potentially to more discriminant embeddings.

The objective of this work is to address the above issues, by allowing the proximity information stored in  $\mathbf{W}$  to accommodate the geometric characteristics of the manifolds, the differing class distributions and their interrelationships. This is possible by incorporating heterogeneously calculated proximity information for the different blocks of  $\mathbf{W}$ . A fairly general model definition for such a *composite weight matrix* is

$$w_{ij} = \begin{cases} f_l(\mathbf{x}_i, \mathbf{x}_j) & \text{if } y_i = y_j = l \wedge \mathbf{x}_j \in N_F(\mathbf{x}_i, k_l) \wedge \\ & \mathbf{x}_i \in N_F(\mathbf{x}_j, k_l), \\ -g_{pq}(\mathbf{x}_i, \mathbf{x}_j) & \text{if } y_i = p \wedge \mathbf{x}_j \in N_E(\mathbf{x}_i, k_{pq}, q) \wedge \\ & y_j = q \wedge \mathbf{x}_i \in N_E(\mathbf{x}_j, k_{pq}, p), \\ 0 & \text{otherwise.} \end{cases} \quad (6.10)$$

In Eq. (6.10), the weight  $w_{ij}$  for friends is controlled independently for each  $l^{\text{th}}$  diagonal block by a different similarity measure  $f_l(\mathbf{x}_i, \mathbf{x}_j)$ . To selectively enforce neighborhood localization, friendship is restricted to pairs of  $\mathbf{x}_i$  and  $\mathbf{x}_j$  that are mutually the  $k_l$ -nearest friends of each other. The model allows for a different  $f_l(\cdot, \cdot)$  and  $k_l$  for each class  $l \in \{1, \dots, c\}$ . For the enemies, an individual similarity measure  $g_{pq}(\cdot, \cdot)$  is defined for each class pair  $(p, q) \in \{1, \dots, c\}^2$  with  $p \neq q$ . For simplicity here symmetry is assumed, that is  $g_{pq} = g_{qp}$ . For neighborhood control, pairwise enmity between  $\mathbf{x}_i$  and  $\mathbf{x}_j$  is defined only for samples mutually appearing within the  $k_{pq}$ -nearest enemies of each other. The set  $N_E(\mathbf{x}_i, k_{pq}, q)$  denotes the  $k_{pq}$ -nearest enemies of  $\mathbf{x}_i$  from class  $q$ , and its formation is based on a search with the proximity function  $g_{pq}(\cdot, \cdot)$ .

Using this model definition, the similarity matrix  $\mathbf{W}$  can now be expressed (assuming a simultaneous re-ordering of its rows and columns, to keep intra-class samples

together in the form of blocks), as

$$\mathbf{W} = \mathbf{F} - \mathbf{G} = \begin{bmatrix} \mathbf{F}_1 & -\mathbf{G}_{12} & -\mathbf{G}_{13} & \dots & -\mathbf{G}_{1c} \\ -\mathbf{G}_{12}^T & \mathbf{F}_2 & -\mathbf{G}_{23} & \dots & -\mathbf{G}_{2c} \\ -\mathbf{G}_{13}^T & -\mathbf{G}_{23}^T & \mathbf{F}_3 & \dots & -\mathbf{G}_{3c} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\mathbf{G}_{1c}^T & -\mathbf{G}_{2c}^T & -\mathbf{G}_{3c}^T & \dots & \mathbf{F}_c \end{bmatrix}. \quad (6.11)$$

Each block  $\mathbf{F}_l \in \mathbb{R}^{n_l \times n_l}$  contains the pairwise similarities between samples from the  $l^{\text{th}}$  class, and each off-diagonal block  $\mathbf{G}_{pq} \in \mathbb{R}^{n_p \times n_q}$  holds the pairwise similarities of samples from the  $p^{\text{th}}$  and the  $q^{\text{th}}$  classes.  $\mathbf{F}$  and  $\mathbf{G}$  are the overall friend and enemy matrices. All  $\mathbf{F}_l$  and  $\mathbf{G}_{pq}$  blocks can be sparse depending on  $k_l$  and  $k_{pq}$ , and are symmetric as mutual friends and enemies are considered.

The identification of this model requires the fine-tuning of the positive integer parameters  $k_l$ ,  $k_{pq}$ , and the similarity functions  $f_l(\cdot, \cdot)$  and  $g_{pq}(\cdot, \cdot)$ . In the form proposed by Eq. (6.10), there are  $c$   $k_l$  and  $f_l(\cdot, \cdot)$  parameters, while  $c(c-1)/2$   $k_{pq}$  and  $g_{pq}(\cdot, \cdot)$  ones for the enemy weights. For problems with very large number of classes, the model can be restricted to the use of  $c$  enemy blocks, by having  $g_{pq} = g_{pr}$  and  $k_{pq} = k_{pr}$ ,  $\forall r \neq q$ . In this case, only  $c$  integer  $k_{pq}$  and function  $g_{pq}(\cdot, \cdot)$  parameters are needed. The form of Eqs. (6.10), (6.11) is generic and can represent a large type of heterogeneous proximity models. For example, assuming a single  $k_l$  and  $f_l(\cdot, \cdot)$  for all  $c$  classes and a single  $k_{pq}$  and  $g_{pq}(\cdot, \cdot)$  for all enemy pairs, other existing homogeneous proximity models can be obtained (with comparable neighborhood control), such as DNE, OLPP-R, SONPP, etc. The mechanism proposed for the identification of the model of Eq. (6.10), is described in detail in Section 6.2.

To generate the optimal embeddings  $\mathbf{Z}$ , Eq. (6.2) is solved subject to different orthogonality constraints on  $\mathbf{Z}$ , in order to keep the embedding coordinates different. In this work, the embeddings are expressed as linear combinations of the input features  $\mathbf{X}$ , following [74], via the transformation  $\mathbf{Z} = \mathbf{X}\mathbf{P}$ , where  $\mathbf{P}$  is the  $m \times b$  projection matrix. Finally, to flexibly utilize the proximity information stored in  $\mathbf{W} = \mathbf{F} - \mathbf{G}$ , the distances between friends are minimised, while the distances between enemies are maximised in the embedding space, by obtaining the optimal projections

$$\mathbf{P}^* = \underset{\substack{\mathbf{P} \in \mathbb{R}^{m \times b}, \\ \mathbf{P}^T (\mathbf{X}^T \tilde{\mathbf{G}} \mathbf{X} + \lambda \mathbf{I}_m) \mathbf{P} = \mathbf{I}_b}}{\operatorname{argmin}} \operatorname{tr} \left[ \mathbf{P}^T \mathbf{X}^T \tilde{\mathbf{F}} \mathbf{X} \mathbf{P} \right], \quad (6.12)$$

where  $\tilde{\mathbf{F}} = \mathbf{D}(\mathbf{F}) - \mathbf{F}$  and  $\tilde{\mathbf{G}} = \mathbf{D}(\mathbf{G}) - \mathbf{G}$  are the Laplacian forms of the friend and enemy proximity matrices. The parameter  $\lambda$  is a regularization parameter acting as a mechanism for controlling the degree of importance of the enemy information. For example, for  $\lambda = 0$ , the constraint of Eq. (6.12) assumes the standard form  $\mathbf{Z}^T \tilde{\mathbf{G}} \mathbf{Z} = \mathbf{I}_{b \times b}$  [77], which enforces orthogonality of  $\mathbf{Z}$  with respect to  $\tilde{\mathbf{G}}$ . When  $\lambda \rightarrow \infty$

the constraint becomes  $\mathbf{P}^T \mathbf{P} = \mathbf{I}_{b \times b}$  [72], which enforces orthogonality of the projections. The above optimization can be directly solved using generalized eigenvalue decomposition involving the two matrix expressions in the objective and the constraint of Eq. (6.12).

## 6.2 Evolutionary Induction

The proposed composite weighting matrix reduces the design of an SSE method to the modelling of a set of suitable similarity functions that correctly describe the underlying data structure. With no prior knowledge of the data distribution, and due to the dimensionality of the problem, it is hard to suggest an adequate metric set. However, such similarity functions can be inferred from the dataset itself. A simple approach to model learning is given by grid search, where for a given set of  $N_d$  similarity metrics, and a classification problem with  $c$  class labels,  $N_d^{c(c+1)/2}$  possible combinations need to be tested as building blocks of the composite weighting matrix. Additionally, parameter fine-tuning needs to be done for each possible combination. Such process turns intractable as the number of classes and available similarity metrics increase. Evolutionary algorithms have been used as a way to avoid the computationally burden of direct linear search for automatic design of classification systems [54, 187, 194, 203, 205, 226–228]. In particular, GP has proven to be a suitable tool for classifier induction [229–232], feature extraction [191, 233, 234], and feature selection [205]. The powerful expressive ability of GP allows it to represent different classification models such as decision trees [235], classification rules [236], artificial neural networks [237] and many more. In the present work, GP is used to encode potential SSE methods into a hybrid chromosome, designed to mimic the structure of the composite weighting matrix proposed in the previous section. Beside providing a suitable representation mechanism, GP is used to pose the problem of automatic design of SSE methods as an inference task, where potential models are proposed, evaluated and systematically modified to improve their quality. This work refers to the foregoing inference process as *model induction*. It resembles model selection in the sense that it picks the best performing model out of a set of potential candidates, however model induction has the ability to generate new models if the existing ones perform poorly. This flexibility is granted by the breeding mechanism implemented through the genetic operators.

### 6.2.1 Chromosome Encoding

As previously mentioned, the constitution of each individual in the genetic population is given by a hybrid chromosome designed to accommodate the structure of the matrix shown in Eq. (6.11), which allows each block to be computed separately. Taking advantage of the class information the learning set can be split into  $c$  mutually exclusive

Table 6.1: Example of the pairwise and matrix forms for various existing distance metrics (where  $\circ$  denotes the Hadamard product). The  $t_P$  and  $t_M$  columns are the corresponding times (seconds) taken to calculate all  $n \times n$  distances of all sample pairs from a dataset with  $n = 1,000$  samples and  $m = 20$  features.

	Metric definition		$t_P$	$t_M$
	Pairwise	Matrix		
Cosine norm	$\frac{\mathbf{x}_i^T \mathbf{x}_j}{\ \mathbf{x}_i\ _2 \ \mathbf{x}_j\ _2}$	$\mathbf{A}(\mathbf{X}\mathbf{X}^T)\mathbf{A}$ $\mathbf{A} = (\mathbf{I} \circ (\mathbf{X}\mathbf{X}^T))^{-\frac{1}{2}}$	17.60	0.06
Correlation	$\frac{\mathbf{x}_i^T \mathbf{C}_m \mathbf{x}_j}{\ \mathbf{C}_m \mathbf{x}_i\ _2 \ \mathbf{C}_m \mathbf{x}_j\ _2}$	$\mathbf{A}(\mathbf{X}\mathbf{C}_m\mathbf{X}^T)\mathbf{A}$ $\mathbf{A} = (\mathbf{I} \circ (\mathbf{X}\mathbf{C}_m\mathbf{X}^T))^{-\frac{1}{2}}$ $\mathbf{C}_m = \mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^T$	21.51	0.05
Kullback-Leibler divergence	$(\mathbf{x}_i - \mathbf{x}_j)^T \log_2(\mathbf{x}_i)$ $+ (\mathbf{x}_j - \mathbf{x}_i)^T \log_2(\mathbf{x}_j)$	$\mathbf{A} + \mathbf{A}^T - (\mathbf{B} + \mathbf{B}^T)$ , $\mathbf{A} = (\mathbf{B} \circ \mathbf{I})\mathbf{1}_n\mathbf{1}_n^T$ , $\mathbf{B} = \mathbf{X} \log_2(\mathbf{X}^T)$	39.59	0.15
Euclidean squared	$\ \mathbf{x}_i - \mathbf{x}_j\ _2^2$	$\mathbf{A} + \mathbf{A}^T - 2\mathbf{X}\mathbf{X}^T$ $\mathbf{A} = (\mathbf{X}\mathbf{X}^T \circ \mathbf{I})\mathbf{1}_n\mathbf{1}_n^T$	12.28	0.08

sets  $\mathbf{X}_l = \{\mathbf{x}_j : y_j = l\}_{l=1,2,\dots,c}$ . Given the set  $\mathbf{X}_l$  with  $n_l$  points, the block  $\mathbf{F}_l$  can be computed using two mapping functions. The first one, which is called pairwise representation, uses the mapping  $f_l : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ . The second mapping takes the form  $\mathcal{F}_l : \mathbb{R}^{n_l \times m} \times \mathbb{R}^{n_l \times m} \rightarrow \mathbb{R}^{n_l \times n_l}$ , and will be called matrix representation. Pairwise representation is simple to define, but needs to be evaluated  $n_l \times n_l$  times to compute its corresponding block matrix  $\mathbf{F}_l$ , while matrix representation requires only one evaluation. Additionally, as can be seen from Table 6.1, the proposed matrix representation allows the expression of classic similarity metrics with only a few functions. Thus, beside speeding up computations, it simplifies the function set design as explained in the next section.

Although computationally faster, the use of matrix representation in GP involves the inclusion of syntactic rules governing the construction of potential models [238]. Strongly typed GP (STGP) is a GP variation where each terminal has an assigned data type, and every function has a return type determined by its arguments. To benefit from the advantages of matrix representation, STGP is used to impose syntactic constraints to the learning problem through a robust function and terminal sets, which were specifically designed to grant the inductive engine with a powerful lexicon, able to not only express any existing SSE, but to create new ones. Hence, under the proposed representation, each individual in the GP population is encoded by an hybrid chromo-

some  $[\{\mathcal{F}_l, k_l\}, \{\mathcal{G}_{pq}, k_{pq}\}, \lambda]$ , which comprises of two sets of parse-tree/scalar pairs, and an independent scalar  $\lambda$ . Each pair encodes a similarity metric in matrix representation, and its corresponding neighbourhood size. The set  $\{\mathcal{F}_l, k_l\}_{l=1,2,\dots,c}$ , related to friends connectivity for each class, is evaluated to build the on-diagonal blocks in Eq. (6.11), while the set  $\{\mathcal{G}_{pq}, k_{pq}\}$  consists of  $c(c-1)/2$  pairs defining the off-diagonal blocks.

The selected hybrid chromosome required the design of suitable crossover and mutation operators. Two types of crossover were designed to keep model coherence. `OnePointXover` is a simple one point crossover, where the trees representing each matrix function in the sets  $\{\mathcal{F}_l\}$  and  $\{\mathcal{G}_{pq}\}$  are treated as indivisible units, thus all the trees beyond the crossover point in either parent are swapped. `SelectiveXover` randomly selects a tree in the same category for each parent, then it swaps randomly selected branches with the same return type. The corresponding scalars for each tree are swapped accordingly for both crossover operators. Mutation randomly selects a tree from any category and then it replaces an aleatory branch with a newly generated tree with the same return type. The scalar part of the selected pair is replaced by a random integer bounded between  $[1, \min(n_q, n_q)]$  or between  $[1, n_l - 1]$ , if the tree is in the set  $\{\mathcal{G}_{pq}\}$  or in the set  $\{\mathcal{F}_l\}$ . The optimization of the regularization parameter  $\lambda$  is performed as well by the evolutionary process by randomly changing its value every time a tree mutation is performed.

### 6.2.2 Fitness Function

The evolutionary nature of the proposed inducer requires a suitable mechanism to assess the quality of the individuals in the population. In this work, out-of-sample estimation error via cross-validation (CV) is adopted as fitness function, consequently the fittest individual will exhibit high classification performance. Thus the induction process can be modelled as follows. Given a family of potential models, a classifier  $\varphi$ , and a finite amount of learning data  $\Upsilon = \Omega_i \cup \tilde{\Omega}_i$ , split into training  $\Omega_i$  and validation  $\tilde{\Omega}_i$  sets, the optimal model  $[\{\mathcal{F}_l^*, k_l^*\}, \{\mathcal{G}_{pq}^*, k_{pq}^*\}, \lambda^*]$  is given by the solution to the bi-level optimisation problem

$$\begin{aligned} \min J([\{\mathcal{F}_l, k_l\}, \{\mathcal{G}_{pq}, k_{pq}\}, \lambda]; \Upsilon) &= \sum_{i=1}^h \sum_{j \in \tilde{\Omega}_i} L(y_j, \varphi_{\Omega_i}(\mathbf{x}_j^T \mathbf{P}^*)) \\ \text{s.t. } \mathbf{P}^* &= \underset{\substack{\mathbf{X}=\{\mathbf{x}_k | k \in \Omega_i\}, \\ \mathbf{P} \in \mathbb{R}^{m \times b}, \mathbf{P}^T(\mathbf{X}^T \tilde{\mathbf{G}} \mathbf{X} + \lambda \mathbf{I}_m) \mathbf{P} = \mathbf{I}_b}}{\text{argmin}} \text{tr} [\mathbf{P}^T \mathbf{X}^T \tilde{\mathbf{F}} \mathbf{X} \mathbf{P}] \\ & k_{pq}, k_l \in \mathbb{N}^+, \lambda > 0 \end{aligned} \tag{6.13}$$

where  $L(\cdot)$  is the binary loss function between the label  $y_i$  and the prediction made by the classifier  $\varphi_{\Omega_i}$ .

For the  $j^{\text{th}}$  individual  $[\{\mathcal{F}_l^{(j)}, k_l^{(j)}\}, \{\mathcal{G}_{pq}^{(j)}, k_{pq}^{(j)}\}, \lambda^{(j)}]$  in the GP population, Eq. (6.13) computes the corresponding fitness function  $J(\cdot)$  step by step as follows: (1) Determine the optimal projection matrix  $\mathbf{P}^*$  for the  $j^{\text{th}}$  individual by solving the trace optimisation problem described Eq. (6.12), using the training set  $\Omega_i$ . (2) Compute

the projected features for both the training and validation sets  $\Omega_i$  and  $\bar{\Omega}_i$  with the obtained projection matrix  $\mathbf{P}^*$ . (3) Perform classification using the projected features. (4) Repeat the process (1) to (3) over each split of a  $h$ -fold cross validation ( $h$ -CV) scheme ( $i = 1, 2, \dots, h$ ), and the averaged classification error over the  $h$  groups of validation sets  $\{\bar{\Omega}_i\}_{i=1}^h$  is used as the fitness function for the  $j^{th}$  individual.

### 6.2.3 Function and Terminals

In order to perform a fair model selection, the proposed evolutionary inducer must have the capacity to generate any of the existing SSE methods. This depends on the proposed function and terminal sets, as they define the syntactic rules and vocabulary available to the genetic search. As previously mentioned, one of the reasons for adopting a matrix representation is that it considerably simplifies the function set design, as only a few functions are needed to express classic similarity metrics. Let  $\mathbf{A}$  and  $\mathbf{B}$  be input matrices to any member of the function set, and  $\mathbf{C}$  its output matrix. The basic function set needed to express all the similarity metrics in Table 6.1 is composed of arithmetic functions like addition, subtraction and multiplication, plus five specialised functions: Hadamard product defined as  $\mathbf{C} = \mathbf{A} \circ \mathbf{B} \leftrightarrow c_{ij} = a_{ij}b_{ij}$ ; function `centre`, which generates a  $m \times m$  centring matrix for a given  $n \times m$  input matrix; function `invsqrt`, which delivers a diagonal matrix with  $c_{ii} = a_{ii}^{-1/2}$  on its diagonal; function `ones`, which allows to create variable size matrices with all their elements equal to one; and function `plog2`, which transforms each  $m$ -dimensional sample vector  $\mathbf{x}_i$  into a discrete probability vector  $\mathbf{c}_i$ . To complement the set of arithmetic functions, left and right division were added, which implement right and left multiplication by the inverse (or pseudo-inverse) of its first and second argument, respectively; Hadamard division, defined as  $\mathbf{C} = \mathbf{A} \oslash \mathbf{B} \leftrightarrow c_{ij} = a_{ij}/b_{ij}$ ; and matrix transpose.

As most of the SSE methods are based on laplacian eigenmaps (LE) [71] or on locality linear embedding (LLE) [70], three more specialised functions were considered: function `sdiag` generates a diagonal matrix with its  $i^{th}$  element equal to  $c_{ii} = \sum_j a_{ij}$ , thus providing a mechanism to create Laplacian style matrices; function `recoef` uses its input argument to generate LLE style regression coefficient matrices; and function `exp` provides a way to generate matrices with the same properties as the Gaussian kernel. The terminal set (Table 6.3) was indirectly defined by the function set, and consists of the training data  $\mathbf{X}$  plus a few numeric constants. The complete function set, along with the symbol representing each member can be found in Table 6.2. The expressive power of the proposed function set is showed in Table 6.4, where selected SSE methods are expressed as similarities/dissimilarities matrix functions, using the proposed function and terminal sets. With this representation, similarities between the different methods can be drawn easily. For instance SONPP, DONPP and ONPP share the same expression to characterise similarities between friends, while they differ in the



Table 6.2: Function set. The last column indicates the grammar for each member, where each rule is given as  $[\text{arg}_1 : \text{out}_1]$  for a one-input/one-output function, and as  $[(\text{arg}_1, \text{arg}_2) : \text{out}_1]$  for a two-input/one-output function.  $\mathbf{C}$  is an output matrix built with the evaluation of a given function member with  $\mathbf{A}$  and/or  $\mathbf{B}$  as input matrices.  $\mathbf{1}_n$  refers to the column vector of size  $n$  with all its elements equal to one.  $\mathbf{A}^\dagger$  refers to the pseudo-inverse of matrix  $\mathbf{A}$ .

Symbol	Arity	Description	Input/Output type
$+, -, \circ, \oslash$	2	Matrix addition and subtraction, Hadamard product and division	$[(f,f):f], [(g,g):g], [(h,h):h], [(o,o):o], [(r,r):r], [(s,s):s], [(t,t):t], [(u,u):u]$
$\times$	2	Matrix multiplication	$[\{(h,f),(f,o),(r,u)\}:f], [\{(g,h),(o,g),(s,t)\}:g], [\{(f,g),(r,t),(h,h)\}:h], [\{(g,f),(s,u)\}:o], [\{(f,s),(h,r)\}:r], [\{(g,r),(o,s)\}:s]$
$/$	2	Right matrix division. If $\mathbf{B}$ is square $\mathbf{C} = \mathbf{A}/\mathbf{B} = \mathbf{A}\mathbf{B}^{-1}$ $\mathbf{C} = \mathbf{A}\mathbf{B}^\dagger$ , otherwise	$[(f,o):f], [(g,h):g], [\{(f,f),(h,b),(h,c)\}:h], [(g,g):o], [\{(f,u),(h,t)\}:r], [\{(g,t),(o,u)\}:s]$
$\backslash$	2	Left matrix division. If $\mathbf{A}$ is square $\mathbf{C} = \mathbf{A}\backslash\mathbf{B} = \mathbf{A}^{-1}\mathbf{B}$ $\mathbf{C} = \mathbf{A}^\dagger\mathbf{B}$ , otherwise	$[(h,f):f], [(o,g):g], [(g,g):h], [(f,f),(o,d):o], [\{(g,s),(h,r)\}:r], [\{(f,r),(o,s)\}:s]$
$(\cdot)^T$	1	Matrix transpose	$[g:f], [f:g], [h:h], [o:o], [r:t], [s:u]$
ones	2	Variable size all-one matrix $\mathbf{C} = \mathbf{1}_n \mathbf{1}_m^T$ $\mathbf{C} \in \mathbb{R}^{n \times m}$	$[(a,d):u], [(b,b):h], [(d,d):o], [(b,a):r], [(d,a):s]$
sdiag	1	If $\mathbf{A} \in \mathbb{R}^n \times \mathbb{R}^m$ , $\mathbf{C} = \mathbf{I} \circ (\mathbf{A} \mathbf{1}_m \mathbf{1}_n^T)$ , if $\mathbf{A} \in \mathbb{R}^n$ , $\mathbf{C} = \mathbf{I} \circ (\mathbf{A} \mathbf{1}_n^T)$	$[\{f,h,r\}:h], [\{g,o,s\}:o]$
pdiag	1	Extracts the diagonal of a given square matrix. $\mathbf{C} = (\mathbf{I} \circ \mathbf{A}) \mathbf{1}_m$	$[hr],[o:s]$

Table 6.2: (continued)

Symbol	Arity	Description	Input/Output type
recoef	1	Regression coeff. matrix [70]	[f:h]
invsqrt	1	$\mathbf{C} = (\mathbf{I} \circ \mathbf{A})^{-1/2}$	[h:h],[o:o]
centre	1	Centring matrix $\mathbf{C} = \mathbf{I} - \frac{1}{m} \mathbf{1}_m \mathbf{1}_m^T$	[f:h],[g:o]
exp	1	Matrix exponential function $c_{ij} = e^{a_{ij}}$	[h:h],[o:o]
plog2	1	$c_{ij} = \log_2 \left( \frac{a_{ij} - \min([a_{ij}])}{(a_i^T - \min([a_{ij}]) \mathbf{1}_m^T) \mathbf{1}_m} \right)$	[f:f],[g:g]

Table 6.3: Terminal set indicating symbol and type used for each terminal along with its description.

Symbol	Type	Description
$1$	a	Constant one
$n_l$	b	Number of points in the block data
$n$	c	Number of points in the training data
$m$	d	Number of dimensions of the training data
$\mathbf{X}$	f	Block data

Table 6.4: Existing homogeneous SSE methods expressed using the employed matrix notation, with single  $F_l$  and single  $G_{pq}$  ( $M_l$  corresponds to the  $l^{th}$  block of the reconstruction coefficient matrix  $\mathbf{M}$  in Eq. (6.7), also  $\mathbf{A} = \mathbf{1}_{n_p} \mathbf{1}_{n_q}^T (\mathbf{I} \circ \mathbf{X}_q^T \mathbf{X}_q)$ ,  $\mathbf{B} = (\mathbf{I} \circ \mathbf{X}_p^T \mathbf{X}_p) \mathbf{1}_{n_p} \mathbf{1}_{n_q}^T$ ,  $\mathbf{C} = 2\mathbf{X}_p^T \mathbf{X}_q$ ).

Method	$F_l$	$G_{pq}$
DNE [83]	$\mathbf{1}_{n_l} \mathbf{1}_{n_l}^T$	$\mathbf{1}_{n_p} \mathbf{1}_{n_q}^T$
MMC [75]	$\frac{2}{n_l} \mathbf{1}_{n_l} \mathbf{1}_{n_l}^T - \frac{1}{n} \mathbf{1}_{n_l} \mathbf{1}_{n_l}^T$	$\frac{1}{n} \mathbf{1}_{n_p} \mathbf{1}_{n_q}^T$
OLPP-R [78]	$\frac{1}{n_l} \mathbf{1}_{n_l} \mathbf{1}_{n_l}^T$	$\beta \mathbf{1}_{n_p} \mathbf{1}_{n_q}^T \oslash (\sigma + (\mathbf{A} + \mathbf{B} - 2\mathbf{C}) \oslash (\mathbf{A} + \mathbf{B}))$
SONPP [72]	$\mathbf{M}_l + \mathbf{M}_l^T - \mathbf{M}_l^T \mathbf{M}_l$	$\mathbf{0}$
DONPP [80]	$\mathbf{M}_l + \mathbf{M}_l^T - \mathbf{M}_l^T \mathbf{M}_l$	$\beta \mathbf{1}_{n_p} \mathbf{1}_{n_q}^T$
ONPP-R [78]	$\mathbf{M}_l + \mathbf{M}_l^T - \mathbf{M}_l^T \mathbf{M}_l$	$\beta \mathbf{1}_{n_p} \mathbf{1}_{n_q}^T \oslash (\sigma + (\mathbf{A} + \mathbf{B} - 2\mathbf{C}) \oslash (\mathbf{A} + \mathbf{B}))$

way they characterise the similarities among enemies.

The aforementioned function and terminal sets need a control mechanism to govern the way they are combined when generating new parse trees. Such mechanism is provided by a set of data types, along with a grammar which provides syntactic rules for the generation of trees to ensure they meet appropriate type constraints. The devised grammar can be found in Table 6.2, along with the proposed function set. To avoid overloading each function with argument parsing, it was decided to let STGP manage the problem by defining apparent data type duplicates, as can be seen in Table 6.5. Those apparent type duplicates differ in size, and are used to define a syntactic rule for each case. For instance, the type matrix has four apparent duplicates, allowing to express the behaviour of all the functions for different matrix sizes. Although the function set is mainly composed of simple arithmetic functions, its true power lies in the syntactic rules defined for each member. STGP requires that each terminal has its own data type, so that only certain functions are allowed to receive terminal nodes as arguments. Such data types can be found along the terminal set definition in Table 6.3.

## 6.3 Experimental Results

### 6.3.1 Datasets

The proposed evolutionary framework, that for propose of brevity will be called Evolutionary Embedding Analysis (EEA), was compared against the SSE methods shown in Table 6.4 in the classification task of six, real world datasets summarised in Table 6.6. Such benchmark datasets were downloaded from [239], and partitioned for final model assesment with the help of 10-CV. Specifically, in each fold for each dataset, a

Table 6.5: Types definition along with the symbol used for each type in the grammatical rules.

Symbol	Type
a	Scalar $l$
b	Scalar $n_l$
c	Scalar $n$
d	Scalar $m$
f	Matrix $n \times m$
g	Matrix $m \times n$
h	Matrix $n \times n$
o	Matrix $m \times m$
r	Column $n$ -dimensional vector
s	Column $m$ -dimensional vector
t	Row $n$ -dimensional vector
u	Row $m$ -dimensional vector

Table 6.6: Dataset summary

Dataset	Features	Samples
cancer	9	277
diabetes	8	768
flare-solar	9	1066
german	20	1000
heart	13	270
thyroid	5	215

90% split was used for the learning phase (i.e. learning data  $\Gamma = \{\Omega_i \cup \bar{\Omega}_i\}$  available to the inducer) and the remaining 10% samples  $\bar{\Gamma}$  were used to test the generalization performance of the proposed system. The entire inference task was repeated within each fold of the 10-CV model assessment. All mentioned datasets are benchmark, binary classification problems from application fields such as medical diagnosis (e.g. cancer, diabetes, heart and thyroid), astronomy (e.g. flare-solar), and credit assignment (e.g. german). Among them, flare-solar and german are computationally expensive problems due to its high number of instances, while german and heart can be benefited the most of dimensionality reduction.

### 6.3.2 Experimental Setup

All the experiments were run on an iMac with CPU Intel Core 2 Duo at 2.66 GHz, 4 GB RAM, OS X 10.6.4, and MATLAB 2009b. The GP algorithm was implemented using a modified version of GPTIPS [240]. For the competing SSE methods that require

the definition of a inter- and intra-class neighbourhood, their respective number of neighbours were optimised by a simple evolutionary search implemented with help of a genetic algorithm (GA) run over 30 generations with a population of 10 individuals, cross-over rate of 70% and mutation rate of 30%. Each variable was encoded as a variable-length binary word. The intra-class neighbours were bounded within the range  $[1, n_k - 1]$ , thus the number of bits reserved were  $\lfloor \log_2(n_k) \rfloor$ . Likewise, the inter-class neighbours between class  $p$  and class  $q$  were bounded within the range  $[1, \min(n_p, n_q) - 1]$ , giving a word length of  $\lfloor \log_2(\min(n_p, n_q)) \rfloor$  bits. The number of extracted features  $b$  for the competing algorithms was encoded as well into the chromosome as a four-bit word. Since the objective of the evolutionary search for the competing algorithms is to perform model selection, the selected fitness function for the GA was cross-validation. To perform a fair comparison, parameter fine-tuning was conducted using the same partition scheme as previously described to perform final model assessment. Descriptive statistics for final model assessment are reported in Table 6.7.

### 6.3.3 Experimental Results and Analysis

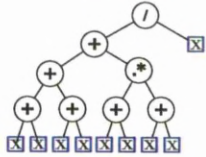
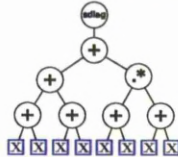










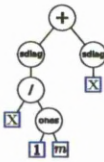
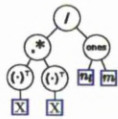
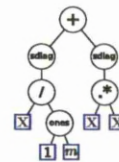

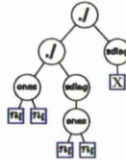

Table 6.7 presents the performance of each selected SSE method and compares them with EEA. As can be seen, EEA presents better generalization performance in all the compared classification tasks. As expected, the classification performance for each benchmark problem varies depending on the competing SSE method. A measure of such variability is given by the average deviation from the ground truth for each competing method, taking EEA misclassification error as ground truth. Using this measure of deviation, the benchmark problems can be ranked in decreasing order as follows: `thyroid`, `flare-solar`, `diabetes`, `german`, `heart`, and `cancer`. Nevertheless, since there is a trade-off between the number of reduced dimensions and classification performance, it is only fair to produce as well an analysis of the predicted number of dimensions for each competing method to completely characterise each problem. In Table 6.7 the optimal number of reduced dimensions learned for each competing SSE problem is also given.

A clear example of the interdependence between dimensionality reduction and classification performance is illustrated by the `cancer` dataset, which ranked last due to the high variability in the classification performance reported by the competing methods, however they agreed most of the time in the number of reduced dimensions, generating an average deviation of  $\pm 2.16$  from the ground truth (i.e.  $d = 2$  predicted by EEA). If the average deviation from the true estimate in the number of reduced dimensions is taken as a measure of the problem complexity, all the selected benchmark problems can be listed in decreasing order of complexity as follows: `flare-solar`, `diabetes`, `german`, `thyroid`, `heart`, and `cancer`. This apparent contradiction, where `cancer` problem can not be correctly classified in the subspace produced by all the competing

Table 6.7: Comparison of the proposed framework including median values of the classification error assessed with 10-CV, and optimal parameters for selected SSE methods consisting of size of the local neighbourhood  $k$ , number of extracted features  $b$ , and penalty parameter  $\beta$ .

Dataset	EEA		MMC		DNE		OLPP-R		SONPP		DONPP		ONPP-R	
	Error	$b/k$	Error	$b$	Error	$b/k$	Error	$b/\beta$	Error	$b$	Error	$b/\beta$	Error	$b/\beta$
cancer	2.99	2/23	4.90	3	4.92	5/48	30.65	5/2.0	45.21	2	5.43	5/1.10	44.93	2/2.0
diabetes	24.48	2/80	24.60	5	24.62	5/9	31.10	5/0.7	37.51	5	27.49	5/1.50	37.11	5/0.2
flare-solar	32.22	2/1	37.36	4	35.75	5/36	38.88	4/1.7	34.29	9	32.65	2/0.80	44.50	4/1.5
german	29.10	2/1	36.62	5	31.00	3/29	40.25	5/1.6	43.11	5	31.66	5/0.70	42.02	5/1.6
heart	14.07	2/1	15.80	5	18.52	5/1	40.89	1/0.2	44.85	1	20.74	5/1.80	43.70	1/1.0
thyroid	11.16	2/1	13.01	2	11.56	4/2	12.83	5/0.1	12.91	4	12.78	5/0.30	12.87	5/0.2

Table 6.8: Best evolved models for each classification problem, along with its tree representation.

cancer			
	$\mathcal{F}_1 = (4X + 2X .* 2X) / X$	$\mathcal{F}_2 = \text{sdiag}(4X + 2X .* 2X)$	$\mathcal{G}_{12} = \text{ones}(n_l, n_l)$
diabetes			
	$\mathcal{F}_1 = XX^T$	$\mathcal{F}_2 = XX^T$	$\mathcal{G}_{12} = XX^T$
flare-solar			
	$\mathcal{F}_1 = \text{ones}(n_l, n_l)$	$\mathcal{F}_2 = \text{ones}(n_l, n_l)$	$\mathcal{G}_{12} = \text{sdiag}(\text{ones}(n_l, 1))$
german			
	$\mathcal{F}_1 = \text{ones}(n_l, n_l)$	$\mathcal{F}_2 = XX^T$	$\mathcal{G}_{12} = \text{sdiag}(X)$
heart			
	$\mathcal{F}_1 = \text{sdiag}(X / \text{ones}(1, m)) + \text{sdiag}(X)$	$\mathcal{F}_2 = (X^T .* X^T) \setminus \text{ones}(m, n_l)$	$\mathcal{G}_{12} = \text{sdiag}(X / \text{ones}(1, m)) + \text{sdiag}(X .* X)$
thyroid			
	$\mathcal{F}_1 = \text{ones}(n_l, n_l)$	$\mathcal{F}_2 = \text{ones}(n_l, n_l) ./ \text{sdiag}(\text{ones}(n_l, n_l)) ./ \text{sdiag}(X)$	$\mathcal{G}_{12} = \text{ones}(n_l, n_l)$

SSE methods even though its deviation from ground truth in the number of reduced dimensions is low, can be easily explained by the *no free lunch theorem*. Clearly MMC, DNE, and DONPP are more suitable for this problem than the rest of the competing SSE methods.

Although most of the time the competing methods do not agree on the number of reduced dimensions, this situation is completely different for diabetes dataset, as all the selected SSE methods produce the same value for  $d$ . This could mislead the reader to think  $d = 5$  is the true value for such quantity, however this hypothesis can be easily discarded by noting that EEA exhibits the lowest classification error among the competing methods. Besides showing the lowest misclassification error, EEA also shows the lowest number of reduced dimensions for all the benchmark problems. Thus, EEA effectively finds an optimal pareto point where both objectives, misclassification error and number of reduced dimensions, are jointly minimised.

Table 6.8 shows the best evolved models, as tree representation, in EEA for each classification problem. In general a correlation can be observed between the complexity of the model and the number of original features, with german dataset being the only exception. Additionally, most of the evolved models make use of the term  $\text{ones}(n_p, n_p)$ , which represents the binary adjacency matrix, indicating it plays a central role for SSE methods. Bear in mind that every time the term  $\text{sdiag}$  is used as the root node of trees in the set  $\{\mathcal{G}_{pq}\}$ , a zero weight matrix is effectively used since only the off-diagonal elements of matrix  $\mathbf{G}_{pq}$  are needed. For cancer dataset, EEA evolves a model that heavily weighs euclidean norm within the *benign class* (class label 2), as can be seen in Table 6.8, first row, second column.

On the other hand, although it uses a binary adjacency matrix to describe the relationships among enemies, the size of the neighbourhood considered is significant. Another example of complex data structure is given by heart dataset, where a different metric is used to weight the euclidean norm in the embedding subspace, but this time for the class label 1 (absence of heart disease). Contrary to cancer dataset, the neighbourhood considered is relatively small which indicates great overlapping between classes. The evolved model for diabetes highlights the use of the dot product to describe intra-class and inter-class relationships, suggesting both classes reside on the same manifold.



## 6.4 Summary

This chapter introduced a paradigm for feature extraction with a closed analytical solution. Different from projection pursuit, spectral embedding methods find a projection matrix to a subspace where the distances among friends are minimised, while distances among enemies are maximised. The main contribution of this chapter is the definition of a heterogeneous proximity matrix, where the relations between enemies and friends are described through a set of different similarity metrics. Additionally, the evolutionary framework proposed in the last chapters is used as inducer engine to learn the desired set of similarity metrics. Thus, the proposed method finds an ad-hoc definition of similarities for the classification problem at hand such that the misclassification error is minimized in the projected subspace. Thus, the proposed EEA method effectively finds an optimal pareto point, where two apparent contradicting objectives, misclassification error and number of reduced dimensions, are jointly optimised.

## Chapter 7

# Conclusions and Future Work

### 7.1 Conclusions

A human-competitive alternative in the design of projection methods has been presented in this work. The proposed technique extracts discriminative features that successfully assist in the problem of classification. Two representation formalisms for feature extraction have been studied, namely PP and SE methods. For each case, the problem of tailoring a feature extraction method to a given classification problem was modelled as a black-box approach to system identification, where learning of the model structure and estimation of its parameters were tackled by an evolutionary search implemented via genetic programming.

Interesting dynamics were observed within the evolution process as result of the proposed hybrid chromosome. Although diversity in the initial population encouraged a deeper search of the solution space, seeding the population with known solutions, available from the literature, allowed the evolutionary search to converge faster. Once a good structure was discovered (i.e. a projection index leading to discriminative features for PP, or a combination of similarity metrics inducing discriminative subspaces in the case of SE), its phenotype was spread among the population with variations only on the scalar part, controlled by mutation. It was observed that gaussian mutation performed a local search in the vicinity of potential solutions, while uniform mutation promoted a more global search to avoid local minima.

The final outcome of PP is a set of projection basis that maps the original input space into a lower-dimensional feature space, where the information described by the projection index is highlighted. Although previous research has employed genetic algorithms to search for an optimal projection matrix instead of looking for the structure of a projection index [54], such formulation lacks information as why specific features were selected or combined in a determined manner. The proposed method indirectly conducts an exploratory analysis of the data, as individual models are evolved, and yields a model that best describes the information needed to be preserved into the new feature space as to improve classification, giving in this way a new insight to the user.

For the case of linear projections, the proposed evolutionary framework was able to learn an optimal projection index which delivered useful features for classification. The inclusion of a stopping criterion in sequential PP allowed us to determine the number of latent variables in an automated way. For the evolved index, the number of latent variables identified by the stopping criterion was not always lower than those for the standard indices. Nevertheless, experiments confirmed the proposed evolutionary projection pursuit system helps to capture as much structure as possible when performing dimensionality reduction. Additionally, with the proposed function and terminal sets a hybridisation of supervised and unsupervised feature extraction was achieved in the context of projection pursuit, boosting in this way the classification abilities of the proposed system.

It was observed that for the nonlinear extension of PP, the nonlinear classification boundaries a kNN classifier implicitly builds, diminish the effect of the kernel-induced nonlinear extracted features. The proposed method searches for an optimal feature kernel-based projector, which needs to separate the nonlinearly separable data in a way to comply optimally with the classifiers boundaries. It seems that using the kNN, the PP optimisation was not pushed or forced adequately to take the very best out of the ability of the kernel-based PP feature extraction stage. After spending some-time experimenting with different classifiers, it was found that the best choice for the powerful kernel-based PP was the simplest classifier, that is an LDA which bears no hyperparameters and is very easy to train. The rationale is that the linear decision boundaries seemed to encourage better the induction of highly discriminant features in the kernel-induced space. Additionally, although a more complex classifier bears more computational burden, this burden does not guarantee a better performance, as shown in the case of EKPP+kNN and EKPP+LDA. The claim that combining two stages of nonlinearity discrimination is detrimental to classification was further supported by experiments with KLPP. This time LPP provided the first layer of nonlinear discrimination, on top of that the data is previously preprocessed with a kernel mapping function, providing the second layer.

Regarding spectral embedding methods, it was detected that a correlation exist between the complexity of the optimal evolved model and the number of original features, indicating more work needs to be done to reduce high-dimensional datasets. Additionally, it was discovered that binary adjacency matrices play an important role in the performance of SSE methods, as most of the evolved models included at least one such term. The proposed hypothesis of heterogenous manifolds residing in the non-observable space was ratified by experimental results, where only one out of six tested datasets resulted in a homogeneous similarity metric measuring both inter-class and intra-class relationships. Furthermore, with this results it was shown that the proposed evolutionary framework can select when to use single or multiple similarity

metrics.

Although the present work showed to be a human-competitive alternative in the design of projection methods for feature extraction, its greatest drawback, inherited from the use of evolutionary methodologies, is the large learning time taken to induce the desired model. Different techniques such as seeding of the initial population, early stopping, and adaptive mutation rate were successfully implemented, showing a considerable reduction of the learning curve without sacrificing classification performance. For the case of spectral embedding methods, besides applying the aforementioned techniques to the evolutionary loop, employing matrix notation in the evolved similarity metrics represented a considerable saving in processing time. Although a direct implementation in C/C++ could have speeded up the evolutionary process, developing of a robust library in C/C++ with reliable algorithms commonly used in machine learning would have taken more time than available, thus the option to implement and reuse MATLAB code was justified for a better development time.

## 7.2 Future Work

One of the aspects that motivated the present work is the ability of feature extraction to compact information into a set of meaningful features. The framework exposed in this thesis can be adapted to explore instance reduction and prototype abstraction for instance-based learning algorithms. In this case, instead of embedding important information into the feature space, it is the number of samples that will be reduced, trying to preserve the local structure of the data. This work may be useful to reduce the large computational complexity and long response times affecting instance-based learning algorithms such as the nearest neighbour rule. Additionally, since feature extraction and prototype reductions could be explained by the same approach. Simultaneous feature and prototype abstraction could be another interesting topic to explore.

As GP lies at the heart of the present thesis, a number of possible future studies can be carried to improve the training time by developing new genetic operators. Additionally, further information regarding the targeted problem can be encoded into the foregoing operators. For example, in the case of spectral embedding, topological-oriented genetic operators could be designed to enforce evolution of homeomorphic functions that could guarantee preservation of certain geometrical properties in the embedding space. Regarding implementation, further experiments can be carried to explore an efficient genetic programming implementation in a parallel computer or cluster, providing a framework to investigate island models, and interaction of different species with techniques such as migration and selective pressure.

Although the impact of the classifier choice for EKPP was studied, further experiments need to be done to completely characterise the performance of several classifiers with the proposed evolutionary framework. Even better, jointly identification of the

structure not only of the feature extraction stage, but also of the classifier, could be performed by introducing a multigene chromosome, encoding in one or more genes the targeted classifier structure. Nevertheless, one must proceed with caution as the representation formalism of GP restrict the classifier to take one of three models: decision tree, discriminant function, or classification rules. A single instance of these models can discriminate between two classes, but an extra parameter may be needed to control the number of instances needed for a multiclass classification problem. Which lead us to another interesting topic for future research, that is the study of classifier ensembles. With the ability of evolving optimal individual classifiers to tackle a multiclass classification problem, an interesting question to consider would be whether a set of also optimal instances would perform better in combination than that single multiclass classifier.

The present work uses classification accuracy as performance measure to select an optimal model, however a different objective functions could also be used, focusing in other aspects of the model, such as minimum description length or Akaike information criterion. Furthermore, a composite metric could be designed to cover model complexity and classification accuracy such that the evolved model not only presents optimal performance for the selected classification task, but also low complexity in its structure to allow the user better interpretation of the results. Different techniques developed to design and assess classifier ensembles could be extended to help drafting a new fitness function for the proposed evolutionary inducer, focusing in the aforementioned aspects of the model.

Since the current implementation was carried using MATLAB, problems related with memory allocation and computing time can be overcome by using other programming tools such as C/C++ in combination with the aforesaid parallel implementation. The only disadvantage C/C++ programming presents is the lack of a reliable implementation of common linear algebra algorithms. Although projects such as CLAPACK, PLASMA, ScaLAPACK and MAGMA have been recently supported by a herd of developers and increasingly popular among programmers, they still need considerable work to fine tune them to application specific tasks. The same story goes for an out-of-the-shelf GP implementation, where it is difficult to exploit the flexible representation due to C/C++ rigid programming structures. Therefore, it is suggested to approach C/C++ development with caution, as the reader will need to invest a considerable amount of time developing a robust library with reliable algorithms instead of researching the aforementioned topics.

# Bibliography

- [1] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley, 2001.
- [2] K. Fukunaga, *Introduction to statistical pattern recognition*. San Diego, CA: Academic, 1990.
- [3] R. Vigario, J. Särelä, V. Jousmäki, M. Hämmäläinen, and E. Oja, “Independent component approach to the analysis of eeg and meg recordings,” *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 5, pp. 589–593, 2000.
- [4] U. Madhow, “Blind adaptive interference suppression for direct-sequence cdma,” in *Proc. of the IEEE*, vol. 86, no. 10, 1998, pp. 2049–2069.
- [5] A. D. Back and A. S. Weigend, “A first application of independent component analysis to extracting structures from stock returns,” *International Journal of Neural Systems*, vol. 8, no. 4, pp. 473–484, 1997.
- [6] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, Apr. 1997.
- [7] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. London, U.K.: Springer, 2001.
- [8] N. Hansen and A. Ostermeier, “Completely derandomized self-adaptation in evolution strategies,” *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.
- [9] A. Auger and N. Hansen, “Performance evaluation of an advanced local search evolutionary algorithm,” in *Proc. IEEE Congress on Evolutionary Computation*, vol. 2, Sep. 2005, pp. 1777 – 1784.
- [10] A. B. Jimenez, J. L. Lazaro, and J. R. Dorronsoro, “Finding optimal model parameters by deterministic and annealed focused grid search,” *Neurocomputing*, vol. 72, no. 13-15, pp. 2824–2832, 2009.
- [11] C. D. Manning and H. Schütze, *Foundations of statistical natural language processing*. Cambridge, Mass: MIT Press, 1999.

- [12] R. Plamondon and S. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 63–84, Jan. 2000.
- [13] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang, "Face recognition using laplacianfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 3, pp. 328–340, Mar. 2005.
- [14] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, 1991.
- [15] A. Jain, S. Prabhakar, L. Hong, and S. Pankanti, "Filterbank-based fingerprint matching," *IEEE Transactions on Image Processing*, vol. 9, no. 5, pp. 846–859, May 2000.
- [16] A. N. Langville, *Google's PageRank and beyond: the science of search engine rankings*. Princeton, NJ: Princeton Univ. Press, 2006.
- [17] J. Goulermas, A. Findlow, C. Nester, D. Howard, and P. Bowker, "Automated design of robust discriminant analysis classifier for foot pressure lesions using kinematic data," *IEEE Transactions on Biomedical Engineering*, vol. 52, no. 9, pp. 1549–1562, Sep. 2005.
- [18] M. Aung, J. Goulermas, S. Hamdy, and M. Power, "Spatiotemporal visualizations for the measurement of oropharyngeal transit time from videofluoroscopy," *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 2, pp. 432–441, Feb. 2010.
- [19] K.-J. Kim and S.-B. Cho, "An evolutionary algorithm approach to optimal ensemble classifiers for dna microarray data analysis," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 3, pp. 377–388, Jun. 2008.
- [20] Q. Guo, W. Wu, D. Massart, C. Boucon, and S. de Jong, "Feature selection in principal component analysis of analytical data," *Chemometrics and Intelligent Laboratory Systems*, vol. 61, no. 1-2, pp. 123–132, Feb. 2002.
- [21] B. J. M. Webb-Robertson, K. H. Jarman, S. D. Harvey, C. Posse, and B. W. Wright, "An improved optimization algorithm and a bayes termination criterion for sequential projection pursuit," *Chemometrics and Intelligent Laboratory Systems*, vol. 77, no. 1-2, pp. 149–160, 2005.
- [22] Z. Huang, H. C. Chen, C. J. Hsu, W. H. Chen, and S. S. Wu, "Credit analysis with support vector machines and neural networks: A market comparative study," *Decision Support Systems*, vol. 37, no. 4, pp. 543–558, Sep. 2004.

- [23] W. Huang, Y. Nakamori, and S. Y. Wang, "Forecasting stock market movement direction with support vector machine," *Computers & Operations Research*, vol. 32, no. 10, pp. 2513–2522, Oct. 2005.
- [24] J. T. L. Wang, S. Rozen, B. A. Shapiro, D. Shasha, Z. Y. Wang, and M. S. Yin, "New techniques for dna sequence classification," *Journal of Computational Biology*, vol. 6, no. 2, pp. 209–218, 1999.
- [25] L. Shapiro and M. Costa, "Appearance-based 3d object recognition, object representation in computer vision," in *Proc. Int. NSF-ARPA Workshop*, vol. 1, New York, USA, 1994, pp. 51–64.
- [26] S.-S. Chiang, C.-I. Chang, and I. Ginsberg, "Unsupervised target detection in hyperspectral images using projection pursuit," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 39, no. 7, pp. 1380–1391, Jul. 2001.
- [27] A. Ifarraguerri and C.-I. Chang, "Unsupervised hyperspectral image analysis with projection pursuit," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 38, no. 6, pp. 2529–2538, Nov. 2000.
- [28] J. Yang, D. Zhang, J. Yu Yang, and B. Niu, "Globally maximizing, locally minimizing: Unsupervised discriminant projection with applications to face and palm biometrics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 4, pp. 650–664, Apr. 2007.
- [29] W. Jaskowski, K. Krawiec, and B. Wieloch, "Evolving strategy for a probabilistic game of imperfect information using genetic programming," *Genetic Programming and Evolvable Machines*, vol. 9, no. 4, pp. 281–294, Dec. 2008.
- [30] P. Stone, *Intelligent autonomous robots: A robot soccer case study*. San Rafael, CA: Morgan & Claypool, 2007.
- [31] R. Sun, F. Tsung, and L. Qu, "Evolving kernel principal component analysis for fault diagnosis," *Computers & Industrial Engineering*, vol. 53, no. 2, pp. 361–371, Sep. 2007.
- [32] H. Guo, L. Jack, and A. Nandi, "Feature generation using genetic programming with application to fault classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 35, no. 1, pp. 89–99, Feb. 2005.
- [33] T. M. Mitchell, *Machine Learning*. McGraw Hill, 1997.
- [34] G. Hinton and T. J. Sejnowski, *Unsupervised learning and map formation: Foundations of neural computation*. Cambridge, MA, USA: MIT Press, 1999.



- [35] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, 1998.
- [36] V. Vapnik, *Statistical learning theory*. New York, USA: Wiley, 1998.
- [37] A. Y. Ng and M. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *Proc. Conf. Advances in Neural Information Processing Systems*, 2002.
- [38] J. H. Xue and M. Titterington, "Comment on "on discriminative vs. generative classifiers: A comparisson of logistic regression and naive bayes"," *Neural Processing Letters*, vol. 28, no. 3, pp. 169–187, 2008.
- [39] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge, UK: Cambridge University Press, 2000.
- [40] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annual ACM Workshop on Computational Learning Theory*, 1992, pp. 144–152.
- [41] B. Scholkopf, A. J. Smola, R. Williamson, and P. Bartlett, "New support vector algorithms," *Neural Computation*, vol. 12, pp. 1207–1245, 2000.
- [42] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 2nd ed. New York, USA: Academic, 2003.
- [43] C. M. Bishop, *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.
- [44] H. F. Leung, H. K. Lam, S. H. Ling, and K. S. Tam, "Tuning of the structure and parameters of a neural network using an improved genetic algorithm," *IEEE Transactions on Neural Networks*, vol. 14, no. 1, pp. 79–88, Jan. 2003.
- [45] N. Garcia-Pedrajas, C. Hervás-Martínez, and J. Muñoz-Pérez, "Covnet: a cooperative coevolutionary model for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 14, no. 3, pp. 575 – 596, May 2003.
- [46] P. Palmes, T. Hayasaka, and S. Usui, "Mutation-based genetic neural network," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 587–600, May 2005.
- [47] I. T. Jolliffe, *Principal Component Analysis*. New York, USA: Springer-Verlag, 2002.
- [48] Y. Moses, Y. Adini, and S. Ulman, "Face recognition: The problem of compensating for changes in illumination direction," in *Proc. Eur. Conf. on Computer Vision*, no. 286-296, 1994.

- [49] A. Hyvarinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. New York, USA: Wiley, 2001.
- [50] A. Hyvarinen, "Survey on independent component analysis," *Neural Computation Surveys*, vol. 2, no. 1, pp. 94–128, 1999.
- [51] B. Scholkopf, A. Smola, and K. R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998.
- [52] B. Scholkopf and A. Smola, *Learning with kernels: Support vector machines, regularization, optimisation, and beyond*. Cambridge, Mass: MIT Press, 2002.
- [53] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," *Neural Computation*, vol. 12, no. 10, pp. 2385–2404, 2000.
- [54] C. Liu and H. Wechsler, "Evolutionary pursuit and its application to face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 6, pp. 570–582, Jun. 2000.
- [55] L. O. Jimenez-Rodriguez, E. Arzuaga-Cruz, and M. Velez-Reyes, "Unsupervised linear feature-extraction methods and their effects in the classification of high-dimensional data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 2, pp. 469–483, Feb. 2007.
- [56] H. Friedman and J. Tukey, "A projection pursuit algorithm for exploratory data analysis," *IEEE Transactions on Computers*, vol. 23, pp. 881–889, 1974.
- [57] J. Kruskal, "Toward a practical method which helps uncover the structure for a set of multivariate observations by finding the linear transformation which optimize a new index of condensation," in *Statistical Computation*, R. Milton and J. Nelder, Eds. New York, USA: Academic, 1969, pp. 427–440.
- [58] H. Friedman and W. Stuetzie, "Projection pursuit regression," *Journal of the American Statistical Association*, vol. 79, pp. 817–823, 1981.
- [59] H. Friedman, W. Stuetzie, and A. Schroeder, "Projection pursuit density estimation," *Journal of the American Statistical Association*, vol. 79, pp. 599–608, 1984.
- [60] M. Jones and R. Sibson, "What is projection pursuit?" *Journal of the Royal Statistical Society*, vol. 150, no. 1, pp. 1–37, 1987.
- [61] P. Huber, "Projection pursuit," *Annals of Statistics*, vol. 13, no. 2, pp. 435–475, 1985.

- [62] E. Lee, D. Cook, S. Klinke, and T. Lumley, "Projection pursuit for exploratory supervised classification," *Journal of Computational and Graphical Statistics*, vol. 14, no. 4, pp. 831–846, Dec. 2005.
- [63] L. Jimenez and D. Landgrebe, "Hyperspectral data analysis and supervised feature reduction via projection pursuit," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 37, no. 6, pp. 2653–2667, Nov. 1999.
- [64] P. Hall, "On polynomial-based projections indices for exploratory projection pursuit," *Annals of Statistics*, vol. 17, pp. 589–605, 1989.
- [65] D. Cook, A. Buja, and J. Cabrera, "Projection pursuit indices based on expansions with orthonormal functions," *Journal of Computational and Graphical Statistics*, vol. 2, pp. 225–250, 1993.
- [66] E. Lee and D. Cook, "A projection pursuit index for large p small n data," *Statistics and Computing*, vol. 20, no. 3, pp. 381–392, 2010.
- [67] Q. Guo, W. Wu, D. Massart, C. Boucon, and S. de Jong, "Sequential projection pursuit using genetic algorithms for data mining of analytical data," *Analytical Chemistry*, vol. 72, no. 13, pp. 2846–2855, Jul. 2000.
- [68] H. S. Seung and D. D. Lee, "The manifold ways of perception," *Science*, vol. 290, no. 5500, pp. 2268–2269, 2000.
- [69] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.
- [70] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
- [71] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, Jun. 2003.
- [72] E. Kokiopoulou and Y. Saadb, "Orthogonal neighborhood preserving projections: A projection-based dimensionality reduction technique," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 12, pp. 2143–2156, Dec. 2007.
- [73] D. Cai, X. He, J. Han, and H.-J. Zhang, "Orthogonal laplacianfaces for face recognition," *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3608–3614, Nov. 2006.

- [74] X. He and P. Niyogi, "Locality preserving projections," in *Proc. Conf. Advances in Neural Information Processing Systems*. MIT Press, 2003.
- [75] H. Li, T. Jiang, and K. Zhang, "Efficient and robust feature extraction by maximum margin criterion," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 157–165, Jan. 2006.
- [76] M. Sugiyama, "Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis," *Journal of Machine Learning Research*, vol. 8, pp. 1027–1061, 2007.
- [77] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 40–51, Jan. 2007.
- [78] E. Kokopoulou and Y. Saadb, "Enhanced graph-based dimensionality reduction with repulsion laplaceans," *Pattern Recognition*, vol. 42, pp. 2392–2402, 2009.
- [79] S. Zhang, "Enhanced supervised locally linear embedding," *Pattern Recognition Letters*, vol. 30, no. 13, pp. 1208–1218, 2009.
- [80] T. Zhang, K. Huang, X. Li, J. Yang, and D. Tao, "Discriminative orthogonal neighborhood-preserving projections for classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 40, no. 1, pp. 253–263, Feb. 2010.
- [81] T. Zhang, D. Tao, X. Li, and J. Yang, "Patch alignment for dimensionality reduction," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1299–1313, Sep. 2009.
- [82] T. Cox and M. Cox, *Multidimensional scaling*. London, U.K.: Chapman & Hall, 1994.
- [83] W. Zhang, X. Xue, Z. Sun, Y. Guo, and H. Lu, "Optimal dimensionality of metric space for classification," in *Proc. Int. Conf. on Machine Learning*, ACM. New York, USA: ACM, 2007.
- [84] K. Zhang and L.-W. Chan, "Dimension reduction as a deflation method in ICA," *IEEE Signal Processing Letters*, vol. 13, no. 1, pp. 45–48, Jan. 2006.
- [85] R. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton Univ. Press, 1961.

- [86] P. Belhumeur, J. Hespanha, and D. Kriegman, "Eigenfaces vs. fisherfaces: recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, Jul. 1997.
- [87] L. F. Chen, H. Y. M. Liao, M. T. Ko, J. C. Lin, and G. J. Yu, "A new lda-based face recognition system which can solve the small sample size problem," *Pattern Recognition*, vol. 33, no. 10, pp. 1713–1726, Oct. 2000.
- [88] P. Howland, M. Jeon, and H. Park, "Structure preserving dimension reduction for clustered text data based on the generalized singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 25, no. 1, pp. 165–179, 2003.
- [89] D. L. Swets and J. J. Weng, "Using discriminant eigenfeatures for image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 831–836, Aug. 1996.
- [90] J. Ye, "Characterization of a family of algorithms for generalized discriminant analysis on undersample problems," *Journal of Machine Learning Research*, vol. 6, pp. 483–502, 2005.
- [91] H. Yu and J. Yang, "A direct lda algorithm for high-dimensional data with applications to face recognition," *Pattern Recognition*, vol. 34, pp. 206–2070, 2001.
- [92] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Mullers, "Fisher discriminant analysis with kernels," in *Proc. IEEE Signal Processing Society Workshop Neural Networks for Signal Processing IX*, Aug. 1999, pp. 41–48.
- [93] H. Friedman, "Regularized discriminant analysis," *Journal of the American Statistical Association*, vol. 84, no. 405, pp. 165–175, 1981.
- [94] T. Hastie, A. Buja, and R. Tibshirani, "Penalized discriminant analysis," *Annals of Statistics*, vol. 23, no. 1, pp. 73–102, Feb. 1995.
- [95] K. Bennett and E. Parrado-Hernandez, "The interplay of optimization and machine learning research," *Journal of Machine Learning Research*, vol. 7, pp. 1265–1281, 2006.
- [96] L. Xu, "Machine learning problems from optimization perspective," *Journal of Global Optimization*, vol. 47, no. 3, pp. 369–401, 2010.
- [97] G. B. Dantzig, *Linear programming and extensions*. Princeton Univ. Press, 1963.
- [98] S. I. Gass, *Linear programming: Methods and applications*. New York, USA: McGraw Hill, 1960.

- [99] H. A. Taha, *Operation research: An introduction*. New York, USA: Macmillan, 1992.
- [100] J. C. G. Boot, *Quadratic programming*. North-Holland, Amsterdam: North-Holland Pub. Co., 1964.
- [101] G. M. Lee, *Quadratic programming and affine variational inequalities: A qualitative study*. Boston, MA: Springer Science+Business Media Inc., 2005.
- [102] H. W. Kuhn and A. Tucker, "Nonlinear programming," in *Proc. Berkeley Symp. on Math. Stats. and Prob.* Berkeley: University of California Press, 1951.
- [103] J. E. Dennis and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*. Englewood Cliffs, NJ: Prentice Hall, 1983.
- [104] D. J. Wilde, *Optimum seeking methods*. Englewood Cliffs, NJ: Prentice Hall, 1964.
- [105] R. P. Brent, *Algorithms for minimization without derivatives*. Englewood Cliffs, NJ: Prentice Hall, 1983.
- [106] A. L. Cauchy, "Methode generale pur la resolution des systemes d'equations simultanees," *Comptes Rendus de l'Academie de Sciences, Paris*, vol. 25, 1847.
- [107] C. G. Broyden, "Quasi-newton methods and their application to function minimization," *Mathematics of Computation*, vol. 11, no. 2, pp. 431–441, 1967.
- [108] W. C. Davidon, "Variable metric method of minimization," Argonne National Laboratory, Argonne, IL, Report ANL-5990, 1959.
- [109] R. Fletcher, "A new approach to variable metric algorithms," *Computer Journal*, vol. 13, pp. 163–168, 1970.
- [110] G. G. Broyden, "The convergence of a class of double-rank minimization algorithms," *Journal of the Institute of Mathematics and Its Applications*, vol. 6, pp. 76–90, 1970.
- [111] D. Goldfarb, "A family of variable metric methods derived by varational means," *Mathematics of Computation*, vol. 24, pp. 23–26, 1970.
- [112] D. F. Shanno, "Conditioning of quasi-newton methods for function minimization," *Mathematics of Computation*, vol. 24, no. 647-656, 1970.
- [113] M. Tambe, W. L. Johnson, R. M. Jones, F. Koss, J. E. Laird, P. S. Rosenbloom, and K. Schwamb, "Intelligent agents for interactive simulation environments," *AI Magazine*, vol. 16, no. 1, pp. 15–39, 1995.

- [114] J. H. Holland, "Outline for a logical theory of adaptive systems," *Journal of the Association for Computing Machinery*, vol. 3, pp. 297–314, 1962.
- [115] I. Rechenberg, *Cybernetic solution path of an experimental problem*. Farnborough, Hants., U.K.: Royal Aircraft Establishment, Aug. 1965, vol. Library translation No. 1122.
- [116] H. P. Schwefel, "Projekt mhd-stausrahlrohr: Experimentelle optimierung einer zweiphasendüse, teil i," AEG Forschungsinstitut, Berlin, Germany, Tech. Rep. Technischer Bericht 11.034/68,35, Oct. 1968.
- [117] L. J. Fogel, "Autonomus automata," *Industrial Research*, vol. 4, pp. 14–19, 1962.
- [118] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. New York: Oxford University Press, 1996.
- [119] Z. Michaelwicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Germany: Springer, 1996.
- [120] J. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Ann Arbor, MI: The University of Michigan Press, 1975.
- [121] T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds., *Evolutionary Computation 2*. Bristol: Institute of Physics Publishing, 2000.
- [122] K. D. Jong, *Evolutionary computation: A unified approach*. Cambridge, Mass: MIT Press, 2006.
- [123] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Nature Selection*. Cambridge, MA, USA: MIT Press, 1992.
- [124] R. Poli, W. B. Langdon, and N. F. McPhee, *A Field Guide to Genetic Programming*. UK: Lulu Enterprises, 2008.
- [125] W. B. Langdon, T. Soule, R. Poli, and J. A. Foster, *Advances in Genetic Programming 3*. Cambridge, MA, USA: MIT Press, Jun. 1999, no. 8, ch. The evolution of size and shape, pp. 163–190.
- [126] S. Silva and E. Costa, "Dynamic limits for bloat control - variations on size and depth," *Proc. Genetic and Evolutionary Computation Conf.*, vol. 3103, pp. 666–677, 2004.
- [127] S. Pal, S. Bandyopadhyay, and S. Ray, "Evolutionary computation in bioinformatics: a review," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 36, no. 5, pp. 601–615, Sep. 2006.

- [128] P. Ma, K. Chan, X. Yao, and D. Chiu, "An evolutionary clustering algorithm for gene expression microarray data analysis," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 296 – 314, Jun. 2006.
- [129] A. M. Tyrrell and Y. Jin, "Guest editorial: Special issue on evolving developmental systems," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 3, pp. 285 –286, Jun. 2011.
- [130] F. Neri, J. Toivanen, G. Cascella, and Y.-S. Ong, "An adaptive multimeme algorithm for designing hiv multidrug therapies," *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 4, no. 2, pp. 264 –278, Apr. 2007.
- [131] T. Paul and H. Iba, "Prediction of cancer class with majority voting genetic programming classifier using gene expression data," *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 6, no. 2, pp. 353 –367, Apr. 2009.
- [132] K.-S. Leung, K. H. Lee, J.-F. Wang, E. Ng, H. Chan, S. Tsui, T. Mok, P.-H. Tse, and J.-Y. Sung, "Data mining on dna sequences of hepatitis b virus," *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 8, no. 2, pp. 428 –440, Mar. 2011.
- [133] A. Brabazon, M. O'Neill, and I. Dempsey, "An introduction to evolutionary computation in finance," *IEEE Computational Intelligence Magazine*, vol. 3, no. 4, pp. 42 –55, Nov. 2008.
- [134] S. Martinez-Jaramillo and E. Tsang, "An heterogeneous, endogenous and coevolutionary gp-based financial market," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 1, pp. 33 –55, Feb. 2009.
- [135] R. Ruiz-Torrubiano and A. Suarez, "Hybrid approaches and dimensionality reduction for portfolio selection with cardinality constraints," *IEEE Computational Intelligence Magazine*, vol. 5, no. 2, pp. 92 –107, May 2010.
- [136] M. Sternberg and R. Reynolds, "Using cultural algorithms to support re-engineering of rule-based expert systems in dynamic performance environments: a case study in fraud detection," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 4, pp. 225 –243, Nov. 1997.
- [137] W. Sheng, G. Howells, M. Fairhurst, and F. Deravi, "A memetic fingerprint matching algorithm," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 3, pp. 402 –412, Sep. 2007.
- [138] V. Valsalam, J. Bednar, and R. Miikkulainen, "Developing complex systems using evolved pattern generators," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 2, pp. 181 –198, Apr. 2007.



- [139] P. Menon, J. Kim, D. Bates, and I. Postlethwaite, "Clearance of nonlinear flight control laws using hybrid evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 689–699, Dec. 2006.
- [140] M.-H. Hung, L.-S. Shu, S.-J. Ho, S.-F. Hwang, and S.-Y. Ho, "A novel intelligent multiobjective simulated annealing algorithm for designing robust pid controllers," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 38, no. 2, pp. 319–330, Mar. 2008.
- [141] Z. Ma and A. Krings, "Dynamic hybrid fault modeling and extended evolutionary game theory for reliability, survivability and fault tolerance analyses," *IEEE Transactions on Reliability*, vol. 60, no. 1, pp. 180–196, Mar. 2011.
- [142] H. Topcuoglu, B. Demiroz, and M. Kandemir, "Solving the register allocation problem for embedded systems using a hybrid evolutionary algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 5, pp. 620–634, Oct. 2007.
- [143] C. Simons, I. Parmee, and R. Gwynllyw, "Interactive, evolutionary search in upstream object-oriented class design," *IEEE Transactions on Software Engineering*, vol. 36, no. 6, pp. 798–816, Nov. 2010.
- [144] D. White, A. Arcuri, and J. Clark, "Evolutionary improvement of programs," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 4, pp. 515–538, Aug. 2011.
- [145] D. Fogel, "Using evolutionary programming for modeling: an ocean acoustic example," *IEEE Journal of Oceanic Engineering*, vol. 17, no. 4, pp. 333–340, Oct. 1992.
- [146] B. Natarajan, S. Das, and D. Stevens, "An evolutionary approach to designing complex spreading codes for ds-cdma," *IEEE Transactions on Wireless Communications*, vol. 4, no. 5, pp. 2051–2056, Sep. 2005.
- [147] K. Zielinski, P. Weitkemper, R. Laur, and K.-D. Kammeyer, "Optimization of power allocation for interference cancellation with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 1, pp. 128–150, Feb. 2009.
- [148] L.-Y. Tseng and T.-Y. Han, "An evolutionary design method using genetic local search algorithm to obtain broad/dual-band characteristics for circular polarization slot antennas," *IEEE Transactions on Antennas and Propagation*, vol. 58, no. 5, pp. 1449–1456, May 2010.

- [149] M. Ecemis, J. Wikel, C. Bingham, and E. Bonabeau, "A drug candidate design environment using evolutionary computation," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 5, pp. 591–603, Oct. 2008.
- [150] J. Knowles, "Closed-loop evolutionary multiobjective optimization," *IEEE Computational Intelligence Magazine*, vol. 4, no. 3, pp. 77–91, Aug. 2009.
- [151] S. Wong, W. Luo, and K. Chan, "EvoMD: An algorithm for evolutionary molecular design," *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 8, no. 4, pp. 987–1003, Jul. 2011.
- [152] E. Hruschka, R. Campello, A. Freitas, and A. de Carvalho, "A survey of evolutionary algorithms for clustering," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 39, no. 2, pp. 133–155, Mar. 2009.
- [153] P. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40, no. 2, pp. 121–144, Mar. 2010.
- [154] A. Fernández, S. García, J. Luengo, E. Bernado-Mansilla, and F. Herrera, "Genetics-based machine learning for rule induction: State of the art, taxonomy, and comparative study," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 6, pp. 913–941, Dec. 2010.
- [155] R. C. Barros, M. P. Basgalupp, A. C. P. L. F. de Carvalho, and A. A. Freitas, "A survey of evolutionary algorithms for decision-tree induction," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. PP, no. 99, pp. 1–10, 2011.
- [156] K. A. de Jong, W. M. Spears, and D. F. Gordon, "Using genetic algorithms for concept learning," *Machine Learning*, vol. 13, pp. 161–188, 1993.
- [157] J. Furnkranz, "Separate-and-conquer rule learning," *Artificial Intelligence Review*, vol. 13, no. 1, pp. 3–54, Feb. 1999.
- [158] M. Kotani, M. Ochi, S. Ozawa, and K. Akazawa, "Evolutionary discriminant functions using genetic algorithms with variable-length chromosome," in *Proc. Int. Joint Conf. on Neural Networks*, vol. 1, 2001, pp. 761–766 vol.1.
- [159] S. Smith, "Rna search acceleration with genetic algorithm generated decision trees," in *Proc. Int. Conf. on Machine Learning and Applications*, Dec. 2008, pp. 565–570.

- [160] S.-U. Guan and F. Zhu, "An incremental approach to genetic-algorithms-based classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 35, no. 2, pp. 227–239, Apr. 2005.
- [161] Y. Zhang and H. Li, "Linear and nonlinear ica based on mutual information," in *Proc. Int. Sym. on Intelligent Signal Processing and Communication Systems*, 2007, pp. 794–797.
- [162] M. J. Aitkenhead, "A co-evolving decision tree classification method," *Expert Systems With Applications*, vol. 34, no. 1, pp. 18–25, Jan. 2008.
- [163] U. Johansson and L. Niklasson, "Evolving decision trees using oracle guides," in *Proc. IEEE Symp. on Computational Intelligence and Data Mining*, Apr. 2009, pp. 238–244.
- [164] B. Chai, T. Huang, X. Zhuang, Y. Zhao, and J. Sklansky, "Piecewise linear classifiers using binary tree structure and genetic algorithm," *Pattern Recognition*, vol. 29, no. 11, pp. 1905–1917, Nov. 1996.
- [165] K. Palaniappan, F. Zhu, X. Zhuang, Y. Zhao, and A. Blanchard, "Enhanced binary tree genetic algorithm for automatic land cover classification," in *Proc. IEEE Int. Symp. on Geoscience and Remote Sensing*, vol. 2, 2000, pp. 688–692 vol.2.
- [166] M. Bot and W. Langdon, "Application of genetic programming to induction of linear classification trees," *Lecture Notes in Computer Science*, vol. 1802, pp. 247–258, 2000.
- [167] M. Kretowski and M. Grzes, "Global induction of oblique decision trees: An evolutionary approach," in *Intelligent Information Processing and Web Mining*, ser. Advances in Soft Computing, vol. 31, 2005, pp. 309–318.
- [168] T. Kovacs, "Genetics-based machine learning," in *Handbook of Natural Computing: Theory, Experiments and Applications*. Springer-Verlag, 2011.
- [169] A. A. Freitas, *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Berlin, Germany: Springer-Verlag, 2002.
- [170] G. Venturini, "Sia: A supervised inductive algorithm with genetic search for learning attributes based concepts," in *Proc. Eur. Conf. on Machine Learning*, vol. 667, Berlin, Germany, 1993, pp. 280–296.
- [171] J. Aguilar-Ruiz, R. Giraldez, and J. Riquelme, "Natural encoding for evolutionary supervised learning," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 4, pp. 466–479, Aug. 2007.

- [172] K. C. Tan, Q. Yu, and J. H. Ang, "A coevolutionary algorithm for rules discovery in data mining," *International Journal of Systems Science*, vol. 37, no. 12, pp. 835–864, Oct. 2006.
- [173] L. Jiao, J. Liu, and W. Zhong, "An organizational coevolutionary algorithm for classification," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 67–80, Feb. 2006.
- [174] D. P. Greene and S. F. Smith, "Competition-based induction of decision models from examples," *Machine Learning*, vol. 13, pp. 229–257, 1993.
- [175] J.-J. Huang, G.-H. Tzeng, and C.-S. Ong, "Two-stage genetic programming (2SGP) for the credit scoring model," *Applied Mathematics and Computation*, vol. 174, no. 2, pp. 1039–1053, March 2006.
- [176] L. Diosan, A. Rogozan, and J.-P. Pécuchet, "Optimising multiple kernels for SVM by genetic programming," in *Proc. 8th Eur. Conf. on Evolutionary Computation in Combinatorial Optimization*, ser. Lecture Notes in Computer Science, J. I. van Hemert and C. Cotta, Eds., vol. 4972. Naples, Italy: Springer, March 2008, pp. 230–241.
- [177] T. Howley and M. G. Madden, "The genetic kernel support vector machine: description and evaluation," *Artificial Intelligence Review*, vol. 24, no. 3-4, pp. 379–395, November 2005.
- [178] K. Sullivan and S. Luke, "Evolving kernels for support vector machine classification," in *Proc. Conf. on Genetic and Evolutionary Computation*, 2007, pp. 1702–1707.
- [179] T. Phienthrakul and B. Kijisirikul, "GPES: an algorithm for evolving hybrid kernel functions of support vector machines," in *Proc. IEEE Conf. on Evolutionary Computation*, D. Srinivasan and L. Wang, Eds., IEEE. Singapore: IEEE, September 2007, pp. 2636–2643.
- [180] M. Gîrdea and L. Ciortuz, "A hybrid genetic programming and boosting technique for learning kernel functions from training data," in *Proc. 9th Int. Symp. on Symbolic and Numeric Algorithms for Scientific Computing*, V. Negru, T. Jelebelean, D. Petcu, and D. Zaharie, Eds. Timisoara, Romania: IEEE, September 2007, pp. 395–402.
- [181] A. Majid, A. Khan, and A. Mirza, "Intelligent combination of kernels information for improved classification," in *Proc. Int. Conf. on Machine Learning and Applications*, Dec. 2005.

- [182] C. Gagné and M. Parizeau, “Coevolution of nearest neighbor classifiers,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 21, no. 5, pp. 921–946, 2007.
- [183] J. Cano, F. Herrera, and M. Lozano, “Using evolutionary algorithms as instance selection for data reduction in kdd: an experimental study,” *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 6, pp. 561–575, Dec. 2003.
- [184] K. Yu, L. Ji, and X. Zhang, “Kernel nearest neighbor algorithm,” *Neural Processing Letters*, vol. 15, no. 2, pp. 147–156, 2002.
- [185] K. Rao Raghuraj, S. Lakshminarayanan, and K. Tun, “Genetic programming models for classification of data from biological systems,” in *Proc. IEEE Congress on Evolutionary Computation*, Sep. 2007, pp. 4154 –4161.
- [186] L. Kuncheva and L. Jain, “Designing classifier fusion systems by genetic algorithms,” *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 327–336, Nov. 2000.
- [187] M. Rizki, M. Zmuda, and L. Tamburino, “Evolving pattern recognition systems,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 594–609, Dec. 2002.
- [188] N. Garcia-Pedrajas, C. Hervás-Martínez, and D. Ortiz-Boyer, “Cooperative co-evolution of artificial neural network ensembles for pattern classification,” *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 271–302, Jun. 2005.
- [189] H. Guo and A. Nandi, “Breast cancer diagnosis using genetic programming generated feature,” *Pattern Recognition*, vol. 39, no. 5, pp. 980–987, May 2006.
- [190] K. Neshatian and M. Zhang, “Genetic programming and class-wise orthogonal transformation for dimension reduction in classification problems,” in *Proc. 11th Eur. Conf. on Genetic Programming*, ser. Lecture Notes in Computer Science, M. O’Neill, L. Vanneschi, S. Gustafson, A. Esparcia-Alcázar, I. D. Falco, A. D. Cioppa, and E. Tarantino, Eds., vol. 4971. Naples, Italy: Springer, March 2008, pp. 242–253.
- [191] M. Muharram and G. Smith, “Evolutionary constructive induction,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 11, pp. 1518 – 1528, Nov. 2005.
- [192] C.-H. Lin and J.-L. Wu, “Automatic facial feature extraction by genetic algorithms,” *IEEE Transactions on Image Processing*, vol. 8, no. 6, pp. 834 –845, Jun. 1999.

- [193] A. J. Perez-Jimenez and J. C. Perez-Cortes, "Genetic algorithms for linear feature extraction," *Pattern Recognition Letters*, vol. 27, no. 13, pp. 1508–1514, Oct. 2006.
- [194] J. Sherrah, R. Bogner, and B. Bouzerdoum, "Automatic selection of features for classification using genetic programming," in *Proc. Australian and New Zealand Conf. on Intelligent Information Systems*, Nov. 1996, pp. 284–287.
- [195] R. A. Davis, A. J. Charlton, S. Oehlschlager, and J. C. Wilson, "Novel feature selection method for genetic programming using metabolomic  $^1\text{H}$  NMR data," *Chemometrics and Intelligent Laboratory Systems*, vol. 81, no. 1, pp. 50–59, March 2006.
- [196] R. Kohavi and G. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1-2, pp. 273–324, Dec. 1997.
- [197] M. G. Smith and L. Bull, "Genetic programming with a genetic algorithm for feature construction and selection," *Genetic Programming and Evolvable Machines*, vol. 6, no. 3, pp. 265–281, September 2005.
- [198] R. Ramírez and M. Puiggros, "A genetic programming approach to feature selection and classification of instantaneous cognitive states," in *Proceedings of the 2007 EvoWorkshop on Applications of Evolutionary Computing*, ser. Lecture Notes in Computer Science, M. Giacobini, A. Brabazon, S. Cagnoni, G. D. Caro, R. Drechsler, M. Farooq, A. Fink, E. Lutton, P. Machado, S. Minner, M. O'Neill, J. Romero, F. Rothlauf, G. Squillero, H. Takagi, S. Uyar, and S. Yang, Eds., vol. 4448. Springer, April 2007, pp. 311–319.
- [199] R. Sikora and S. Piramuthu, "Framework for efficient feature selection in genetic algorithm based data mining," *European Journal of Operational Research*, vol. 180, no. 2, pp. 723–737, July 2007.
- [200] F. Javed, G. S. H. Chan, A. V. Savkin, P. M. Middleton, P. Malouf, E. Steel, J. Mackie, and N. H. Lovell, "Rbf kernel based support vector regression to estimate the blood volume and heart rate responses during hemodialysis," in *Proc. Annu. Int. Conf. IEEE Engineering in Medicine and Biology Society*, Sep. 2009, pp. 4352–4355.
- [201] M. Raymer, W. Punch, E. Goodman, L. Kuhn, and A. Jain, "Dimensionality reduction using genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 2, pp. 164–171, Jul. 2000.

- [202] J. T. Tsai, J. H. Chou, and T. K. Liu, "Tuning the structure and parameters of a neural network by using hybrid taguchi-genetic algorithm," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 69–80, Jan. 2006.
- [203] D. Muni, N. Pal, and J. Das, "A novel approach to design classifiers using genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 183–196, Apr. 2004.
- [204] N. Y. Nikolaev and H. Iba, "Learning polynomial feedforward neural networks by genetic programming and backpropagation," *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 337–350, Mar. 2003.
- [205] D. Muni, N. Pal, and J. Das, "Genetic programming for simultaneous feature selection and classifier design," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 36, no. 1, pp. 106–117, Feb. 2006.
- [206] K. Bennett, J. Hu, G. Kunapuli, and J. Pang, "Model selection via bilevel optimization," in *Proc. IEEE Int. Joint Conf. on Neural Networks*, Vancouver, Canada, Jul. 2006, pp. 1922–1929.
- [207] A. Renyi, "On measures of information and entropy," in *Proc. Berkeley Symp. on Math. Stats. and Prob.*, Jun. 1961, pp. 547–561.
- [208] G. Nason, "Robust projection indices," *Journal of the Royal Statistical Society*, vol. 63, no. 3, pp. 551–567, 2001.
- [209] W. Mendenhall and T. L. Sincich, *Statistics for Engineering and the Science*. Upper Saddle River, NJ: Pearson Prentice Hall, 2007.
- [210] S. Silva and J. Almeida, "Gplab: A genetic programming toolbox for matlab," in *Proc. Nordic MATLAB Conference*, 2003, pp. 273–278.
- [211] G. Cardillo. (2008) Roc curve: Compute a receiver operating characteristics curve. [Online]. Available: <http://www.mathworks.com/matlabcentral/fileexchange/19950>
- [212] L. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Hoboken, NJ: Wiley, 2004.
- [213] D. C. Youla and H. Webb, "Image restoration by the method of convex projections: Part 1 theory," *IEEE Transactions on Medical Imaging*, vol. 1, no. 2, pp. 81–94, Oct. 1982.
- [214] A. M. Tekalp, M. K. Ozkan, and M. I. Sezan, "High-resolution image reconstruction from lower-resolution image sequence and space-varying image restoration,"

- in *Proc. IEEE Int. Conf. on Accustics, Speech and Signal Processing*, 1992, pp. 169–172.
- [215] C. Fyfe and R. Baddeley, “Non-linear data structure extraction using simple hebbian networks,” *Biological Cybernetics*, vol. 72, no. 6, pp. 533–541, 1995.
- [216] A. Smola, O. Mangasarian, and B. Scholkopf, “Sparse kernel feature analysis,” University of Wiscosin Madison, Technical report 99-04, 1999.
- [217] H. Friedman, “Exploratory projection pursuit,” *Journal of the American Statistical Association*, vol. 82, no. 397, pp. 249–266, Mar. 1987.
- [218] J. Mercer, “Functions of positive and negative type and their connection with the theory of integral equations,” *Philosophical Transactions of the Royal Society of London*, vol. 7, pp. 415–446, 1909.
- [219] Nips’03 feature selection challenge, december 8-13, 2003. [Online]. Available: <http://www.nipsfsc.ecs.soton.ac.uk>
- [220] M. West, C. Blanchette, H. Dressman, E. Huang, S. Ishida, R. Spang, H. Zuzan, J. A. Olson, J. R. Marks, and J. R. Nevins, “Predicting the clinical status of human breast cancer by using gene expression profiles,” in *Proc. National Academy of Sciences*, vol. 98, 2001, pp. 11 462–11 467.
- [221] E. Rodriguez-Martinez, J. Goulermas, T. Mu, and J. Ralph, “Automatic induction of projection pursuit indices,” *IEEE Transactions on Neural Networks*, vol. 21, no. 8, pp. 1281 –1295, Aug. 2010.
- [222] I. Guyon, J. Li, T. Mader, P. A. Pletscher, G. Schneider, and M. Uhr, “Competitive baseline methods set new standards for the nips 2003 feature selection benchmark,” *Pattern Recognition Letters*, vol. 28, no. 12, pp. 1438–1444, Sep. 2007.
- [223] W. K. Wong and H. T. Zhao, “Supervised optimal locality preserving projection,” *Pattern Recognition*, vol. 45, no. 1, pp. 186–197, Jan. 2012.
- [224] Y. Wang, Y. Jiang, Y. Wu, and Z. Zhou, “Spectral clustering on multiple manifolds,” *IEEE Transactions on Neural Networks*, vol. 22, no. 7, pp. 1149 – 1161, Jul. 2011.
- [225] A. Goldberg, X. Zhu, A. Singh, Z. Xu, and R. Nowak, “Multi-manifold semi-supervised learning,” *Journal of Machine Learning Research W&CP*, vol. 5, pp. 169–176, 2009.



- [226] E. Jones, P. Runkle, N. Dasgupta, L. Couchman, and L. Carin, "Genetic algorithm wavelet design for signal classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 8, pp. 890–895, Aug. 2001.
- [227] M. Mignotte, C. Collet, P. Perez, and P. Bouthemy, "Hybrid genetic optimization and statistical model based approach for the classification of shadow shapes in sonar imagery," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 2, pp. 129–141, Feb. 2000.
- [228] H. Escalante, M. Montes, and L. Sucar, "Particle swarm model selection," *Journal of Machine Learning Research*, vol. 10, pp. 405–440, Feb. 2009.
- [229] C.-S. Kuo, T.-P. Hong, and C.-L. Chen, "Applying genetic programming technique in classification trees," *Soft Computing*, vol. 11, no. 12, pp. 1165–1172, 2007.
- [230] M. Zhang, X. Gao, and W. Lou, "A new crossover operator in genetic programming for object classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 5, pp. 1332–1343, Oct. 2007.
- [231] A. L. Garcia-Almanza and E. P. K. Tsang, "Evolving decision rules to predict investment opportunities," *International Journal of Automation and Computing*, vol. 5, no. 1, pp. 22–31, 2008.
- [232] G. Folino, C. Pizzuti, and G. Spezzano, "Training distributed gp ensemble with a selective algorithm based on clustering and pruning for pattern classification," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 4, pp. 458–468, Aug. 2008.
- [233] Y. Lin and B. Bhanu, "Evolutionary feature synthesis for object recognition," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 35, no. 2, pp. 156–171, May 2005.
- [234] L. Guo, D. Rivero, J. Dorado, C. R. Munteanu, and A. Pazos, "Automatic feature extraction using genetic programming: An application to epileptic EEG classification," *Expert Systems With Applications*, vol. 38, no. 8, pp. 10 425–10 436, Aug. 2011.
- [235] H. Zhao, "A multi-objective genetic programming approach to developing pareto optimal decision trees," *Decision Support Systems*, vol. 43, no. 3, pp. 809–826, Apr. 2007.
- [236] C. Bojarczuk, H. Lopes, and A. Freitas, "Genetic programming for knowledge discovery in chest-pain diagnosis," *IEEE Engineering in Medicine and Biology Magazine*, vol. 19, no. 4, pp. 38–44, Jul. 2000.

- [237] M. D. Ritchie, A. A. Motsinger, W. S. Bush, C. S. Coffey, and J. H. Moore, "Genetic programming neural networks: A powerful bioinformatic tool for human genetics," *Applied Soft Computing*, vol. 7, no. 1, pp. 471–479, Jan. 2007.
- [238] D. J. Montana, "Strongly typed genetic programming," *Evolutionary Computation*, vol. 3, pp. 199–230, 1995.
- [239] Friedrich miescher laboratory, max-planck institute. [Online]. Available: <http://www.fml.tuebingen.mpg.de/Members/raetsch/benchmark>
- [240] D. P. Searson, D. E. Leahy, and M. J. Willis, "Gptips: An open source genetic programming tool for multigene symbolic regression," in *Proc. Int. Multi Conf. of Engineers and Computer Scientists*, ser. Lecture Notes in Engineering and Computer Science, Hong Kong, Mar. 2010.