Semi-supervised Margin-based Feature Selection for Classification

Samah Hijazi

THÈSE de doctorat en Cotutelle

Pour obtenir le grade de Docteur délivré par

L'Université du Littoral Côte d'Opale

L'Ecole Doctorale SPI - Université Lille Nord-De-France

et

L'Université Libanaise

L'Ecole Doctorale des Sciences et Technologie

Discipline : Sciences et Technologies de l'Information et de la communication, Traitement du Signal et des Images

Présentée et soutenue publiquement par

Samah HIJAZI

le 20 Décembre 2019

Sélection d'Attributs Basée Marge pour la Classification dans un Contexte Semi-Supervisé

Membre du Jury:

M. **Fadi DORNAIKA**	Professeur de Recherche à l'Université du Pays Basque	Rapporteur
M. **Ali MANSOUR**	Professeur à l'Université ENSTA Bretagne	Rapporteur
M. **Kifah TOUT**	Professeur à l'Université Libanaise	Président
M. **Ghaleb FAOUR**	Directeur de Recherche au CNRS-Liban	Examinateur
Mme. **Marwa EL BOUZ**	Enseignante-Chercheuse à Yncréa-Ouest	Examinateur
M. **Denis HAMAD**	Professeur à l'Université du Littoral Côte d'Opale	Directeur de thèse
M. **Ali KALAKECH**	Professeur à l'Université Libanaise	Directeur de thèse
Mme. **Mariam KALAKECH**	Maître de Conférences à l'Université Libanaise	Encadrante de thèse

Cotutelle PhD THESIS

submitted in partial fulfillment for the degree of Doctor of Philosophy

from the University of the Opal Coast
Doctoral School SPI - University of Lille- North France
and
the Lebanese University
Doctoral School of Science and Technology

Specialty : Information and Communication Sciences and Technologies, Signal and Image Processing

Publicly defended by

Samah HIJAZI

on December 20, 2019

Semi-supervised Margin-based Feature Selection for Classification

Committee Members:

Mr. **Fadi DORNAIKA**	Research Professor at the University of the Basque Country	Reviewer
Mr. **Ali MANSOUR**	Professor at the University of ENSTA Bretagne	Reviewer
Mr. **Kifah TOUT**	Professor at the Lebanese University	President
Mr. **Ghaleb FAOUR**	Research Director at CNRS-Lebanon	Examiner
Mrs. **Marwa EL BOUZ**	Teacher-Researcher at Yncréa-Ouest	Examiner
Mr. **Denis HAMAD**	Professor at the University of the Littoral Opal Coast	Thesis Director
Mr. **Ali KALAKECH**	Professor at the Lebanese University	Thesis Director
Mrs. **Mariam KALAKECH**	Senior Lecturer at the Lebanese University	Thesis Supervisor

Contents

Acknowledgment

First and foremost, praises and thanks to the God, the Almighty, for His showers of blessings throughout my travel and research journey.

Special thanks to the University of the Littoral Opal Coast (ULCO), Agence Universitaire de la Francophonie (AUF), and the National Council For Scientific Research (CNRS-L) for supporting this work with a scholarship grant as a part of ARCUS E2D2 project.

I would like to express my deep and sincere gratitude to my thesis director, Professor Denis HAMAD for giving me the opportunity to do research and providing invaluable guidance throughout this research. I would also like to thank Professor Ali KALAKECH, my thesis co-director, and Dr. Mariam KALAKECH, my thesis supervisor, for their valuable advices during my thesis work. It was a great privilege and honor to work under their guidance.

I would also like to express my special thanks to Professor Fadi DORNAIKA and Professor Ali MANSOUR for their precious time and effort to review my work. Special thanks to Professor Kifah TOUT, Professor Ghaleb FAOUR, and Dr. Marwa El BOUZ as well for examining my thesis.

To my friends and research colleagues, Emna CHEBBI, Hiba AL ASSAAD, Pamela AL ALAM, Aya MOURAD, Ali DARWICH, Rim TRAD, Mohammad HARISSA, Ghidaa BADRAN, Tarek ZAAROUR, Ragheb GHANDOUR, Rasha SHAMSEDDINE, and Mohammad Ali ZAITER, thank you all for your constant support. I am also extremely thankful to the amazing Dr. Vinh Truong HOANG for his efforts and advices. Special thanks to Mahdi BAHSOUN, I wouldn't have known about this opportunity without your help.

To my parents and life coaches, Zakaria HIJAZI and Amal WEHBI, no words can express my feelings of gratitude. Thank you for being there every moment, showering me with love, support, encouragement, sincere prayers, and care. I am extremely grateful for all the sacrifices you have done for making my future brighter. To my sister, Sahar HIJAZI simply thank you for being you. I am extremely lucky to be your little sister.

Final words to my partner and supporter Ali REBAIE. Thank you for your patience and for listening to me every single day, to my smallest and simplest problems, to my fears and worries. Thank you for understanding the long silent nights. I owe it all to you.

Notations

$\boldsymbol{X}$	Data matrix $X = \{\boldsymbol{x}_n\}_{n=1}^{N}$
$\boldsymbol{y}_l$	Data Labels vector
N	Number of data points
N_c	Number of points in class c
F	Number of features
C	Number of classes
$\boldsymbol{x}_n$	n-th data point
y_n	Label of data point $\boldsymbol{x}_n$
$\boldsymbol{x}_{ni}$	Value of n-th data point on i-th feature
$\boldsymbol{A}_i$	i-th feature
$\boldsymbol{A}_{in}$	Value of i-th feature on the n-th data point
$\mu_{\boldsymbol{A}_i}$	Mean of the i-th feature
$\mu^c_{\boldsymbol{A}_i}$	Mean of i-th feature for points of class c
$\boldsymbol{S}$	Similarity matrix
$\boldsymbol{D}$	Degree diagonal matrix
ML	Pairwise must-link constraints set
CL	Pairwise cannot-link constraints set
$\boldsymbol{S}^{ML}$	Similarity matrix defined on ML
$\boldsymbol{S}^{CL}$	Similarity matrix defined on CL
$\boldsymbol{S}^{kn}$	Similarity matrix defined on ML and nearest neighbors
$\boldsymbol{D}^{ML}$	Degree diagonal matrix defined on ML
$\boldsymbol{D}^{CL}$	Degree diagonal matrix defined on CL
$\boldsymbol{D}^{kn}$	Degree diagonal matrix defined on ML and nearest neighbors
$\boldsymbol{L}$	Laplacian matrix with $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{S}$
$\boldsymbol{L}_{sym}$	Normalized symmetric Laplacian matrix
$\boldsymbol{L}_{rw}$	Normalized asymmetric Laplacian matrix
$\boldsymbol{L}^{ML}$	Laplacian matrix with $\boldsymbol{L}^{ML} = \boldsymbol{D}^{ML} - \boldsymbol{S}^{ML}$

$\boldsymbol{L}^{CL}$	Laplacian matrix with $\boldsymbol{L}^{CL} = \boldsymbol{D}^{CL} - \boldsymbol{S}^{CL}$
$\boldsymbol{L}^{kn}$	Laplacian matrix with $\boldsymbol{L}^{kn} = \boldsymbol{D}^{kn} - \boldsymbol{S}^{kn}$
$\boldsymbol{I}$	Identity matrix
$\mathcal{V}$	Set of nodes in a graph
$\mathcal{V}_n$	Graph node corresponding to data point $\boldsymbol{x}_n$
E	Set of edges in a graph
$G = (\mathcal{V}, E)$	Undirected graph constructed over $\boldsymbol{X}$
V_i	Variance Score of a particular feature A_i
LS_i	Laplacian Score of a particular feature A_i
F_i	Fisher Score of a particular feature A_i
$CS1_i$	Constraint Score-1 of a particular feature A_i
$CS2_i$	Constraint Score-2 of a particular feature A_i
$CS3_i$	Constraint Score-3 of a particular feature A_i
$CS4_i$	Constraint Score-4 of a particular feature A_i
CLS_i	Constrained Laplacian Score of a particular feature A_i
K	Number of nearest neighbors
$KNN(\boldsymbol{x}_n)$	Set of the K-nearest neighbors to $\boldsymbol{x}_n$
$\boldsymbol{w}$	Weight vector spanning F features
w_i	Weight of feature A_i
$\boldsymbol{H}(\boldsymbol{x}_n)$	Nearhit of point $\boldsymbol{x}_n$
$\boldsymbol{M}(\boldsymbol{x}_n)$	Nearmiss of point $\boldsymbol{x}_n$
T	Number of iterations
$K\boldsymbol{H}(\boldsymbol{x}_n)$	K-nearhits of point $\boldsymbol{x}_n$
$K\boldsymbol{M}(\boldsymbol{x}_n)$	K-nearmisses of point $\boldsymbol{x}_n$
$K\boldsymbol{M}(\boldsymbol{x}_n, c)$	K-nearmisses of point $\boldsymbol{x}_n$ from different class c
$\rho(\boldsymbol{x}_n)$	Hypothesis-margin of point $\boldsymbol{x}_n$
$\boldsymbol{z}$	Hypothesis-margin vector of length F
$\rho(\boldsymbol{x}_n, \boldsymbol{x}_m)$	Hypothesis-margin of a cannot-link constraint
$\rho((\boldsymbol{x}_n, \boldsymbol{x}_m), \boldsymbol{w})$	Weighted Hypothesis-margin of a cannot-link constraint
$\Delta(\boldsymbol{p}_1, \boldsymbol{p}_2)$	General distance function between any two data points
$\Delta(\boldsymbol{A}_i, \boldsymbol{p}_1, \boldsymbol{p}_2)$	General distance function between any two data points on a specific feature
$\Delta_w(\boldsymbol{p}_1, \boldsymbol{p}_2)$	General weighted distance function between any two data points

Abbreviations

RBAs	Relief-Based Algorithms
Relief-Sc	Relief with Side Constraints
ACS	Active Constraint Selection
RCG	Random Constraint Generation
PACS	Propagation of Actively Selected Constraints
FCRSC	Feature Clustering Relief-Sc
Simba	Iterative Search Margin-Based Algorithm
Simba-Sc	Simba with Side Constraints
MI	Mutual Information
mRMR	Minimum Redundancy-Maximum Relevance
K-NN	*K*- Nearest Neighbor
UCI	University California Irvine
SVM	Support Vector Machines
NB	Naive Bayes
C4.5	Decision Tree
CSi	Constraint Score-i
CLS	Constrained Laplacian Score
FLDA	Fisher Linear Discriminant Analysis
PCA	Principal Component Analysis
MDS	Multidimensional Scaling
LLE	Local Linear Embedding
SPCA	Supervised Principal Component Analysis
t-SNE	t-distributed Stochastic Neighbor Embedding
PCC	Pearson Correlation Coefficient
FCBF	Fast Correlation-Based Filter
SFS	Sequential Forward Selection
SBS	Sequential Backward Selection

STIR	STatistical Inference for Relief
SVDD	Support Vector Data Description
SDP	Semi-Definite Programming
MTPs	Meta-Points
LPP	Locality Preserving Projections
MLPP-CLP	Multiple Locality Preserving Projections with Cluster-based Label Propagation

Introduction

With the rapid growth of modern technologies, the limitless number of computer and Internet applications has caused an exponential increase in the amount of generated data in a variety of domains. For instance, in domains such as social media, healthcare, marketing, bioinformatics and biometrics, the data provided such as image, video, text, voice, gene expression microarrays, and other kinds obtained from social relations and the Internet of Things may not only be huge in terms of the data samples, but also in terms of feature dimensionality. This imposes many challenges on effective and efficient data management.

Therefore, the use of data mining and machine learning techniques becomes a necessity for automatically extracting knowledge and uncovering hidden patterns from data. In fact, according to the contextual knowledge and the way of identifying data patterns, these techniques can be broadly categorized into classification, regression (or prediction), and clustering. Data classification, the problem of identifying to which of a set of pre-defined categories a data point should belong, can model many real-world applications. Indeed, datasets are usually represented by two-dimensional matrices where the rows correspond to data samples and columns correspond to the features characterizing them. In this regard, some of the available features characterizing the data may not provide any useful information or even express noise with respect to a certain relevance evaluation criterion (e.g. class discrimination), others may be correlated or redundant which makes the learning process complex, expensive in terms of storage and computation, ineffective, less generalizable and difficult to interpret.

As a solution, finding *narrower* data matrices that can successfully summarize the original ones can be considered. The latter process is known as dimensionality reduction which is a major step in data pre-processing. It can be mainly applied in two different approaches: feature extraction and feature selection. Feature extraction con-

verts the initial input space into a new one of lower dimensions by combining the original features, thus, changing their meaning. Whereas, feature selection simply chooses a small subset of features that best describes a dataset out of a larger set of candidate features with the aim of producing a lower-dimensional space without any core changes to the meaning of features. Consequently, feature selection costs less than feature extraction in terms of computational cost and model interpretability.

Generally speaking, individual feature selection can be generalized into feature weighting/scoring/ranking, by which, each feature is individually assigned a soft relevance score instead of just a binary one. Indeed, the process by which this score is evaluated changes with the change of contextual knowledge. Therefore, in an unsupervised context where no labeling information is available at all, feature selection methods resort to using data similarity and local discriminative information approaches to measure the ability of features in discriminating data groups. On the other side, in a supervised context where data is fully labeled, feature selection methods assign higher scores to features having high correlation measures with class labels. There also exists a whole family of feature selection algorithms that use a margin-based score in order to evaluate and rank features. These are known as Relief-Based Algorithms (RBAs) and were initially suggested in the supervised context for two-class problems. They assign bigger weights for features that best contribute to enlarge a distance metric called hypothesis-margin. This margin is calculated as the difference between the distance from a point to its nearmiss (nearest point having a different label) and the distance to its nearhit (nearest data point having the same label). RBAs proved to generally perform well regardless of the problem specifications. They have low bias filter algorithms (independent of classifiers), considered relatively fast, capable of detecting feature interaction, robust and noise-tolerant in addition to their ability to capture data local dependencies [1, 2].

However, in many real-world applications, there usually exist few labeled data points and lots of unlabeled ones, by which, both supervised and unsupervised feature selection algorithms cannot fully take advantage of all data points in this scenario. Thus, it was wise to use semi-supervised methods to feat both labeled and unlabeled points. Actually, compared to class labels, pairwise constraints are another type of supervision information that can be acquired more easily. These constraints simply specify whether a pair of data points belongs to the same class (must-link constraint) or to different classes (cannot-link constraint) without specifying the classes themselves. Many constraint scores use these two notions to rank features, however, they

still neglect the information provided by unconstrained and unlabeled data.

This led us to suggest a new framework for semi-supervised constrained margin-based feature selection that handles the two core aspects of feature selection: relevancy and redundancy. It consists of, (1) a constrained margin-based feature selection algorithm that utilizes pairwise cannot-link constraints only and benefits from both the local unlabeled neighborhood of the data points as well as the provided constraints, (2) a method for *actively* selecting constraints based on matrix perturbation theory applied on the similarity matrix in addition to the propagation of these constraints through decomposing the problem into a set of independent label propagation subproblems, and (3) a feature clustering method that combines sparse graph representation of the feature space with margin maximization.

In order to compare the performance of our suggested methods with other score-based supervised, unsupervised and constrained feature selection methods, experiments will be initially carried out on well-known benchmark UCI (University California Irvine [3]) and high-dimensional gene expression datasets.

Contributions

The core of this thesis is semi-supervised constrained feature selection for high-dimensional data. As feature selection aims at finding a small subset of relevant and non-redundant features that can successfully summarize the original feature space, our workflow was to focus first on tackling the problem of constraint-relevant feature selection followed by handling redundancy. Throughout this work our contributions can be summarized as follows:

- We provide a comprehensive and concise literature review of Relief-based Algorithms from their four interpretations, called probabilistic [4], comprehensible [5], mathematical [6], and statistical [7]. We highlight their strengths, limitations, variants and extensions, in addition to representing the original Relief as a margin-based algorithm.

- We suggest the semi-supervised margin-based constrained algorithms Relief-Sc (Relief with Side Constraints) and its robust version ReliefF-Sc. They integrate the modification of hypothesis-margin when used with cannot-link constraints, with the analytical solution of the supervised Relief algorithm from its optimiza-

tion perspective. They utilize cannot-link constraints only to solve a simple convex problem in a closed-form providing a unique solution.

- We suggest an active method for pairwise constraints selection called Active Constraint Selection (ACS). The output of this method is used by Relief-Sc and ReliefF-Sc. ACS is based on the matrix perturbation theory, specifically on the First-Order Eigenvector Perturbation theorem. It systematically chooses which pairs of data are more effective in reducing uncertainty. Accordingly, only these pairs are queried for constraints from human-experts, thus, decreasing human labor-cost and avoiding noisy constraints and any ill interactions between a random constraint set and our objective function. In addition, a method for Propagating these Actively Selected Constraints (PACS) to their neighborhood was also suggested.

- We propose extending our semi-supervised feature selection method into a novel combination of feature clustering and hypothesis margin maximization. This method, called Feature Clustering ReliefF-Sc (FCRSC), aims to allow redundancy elimination as part of our overall suggested framework.

Structure of thesis

This thesis is structured as follows:

In the first chapter, we introduce the definitions of dimensionality reduction, feature extraction, feature selection, feature relevancy, and feature redundancy. We also present the main data notations and knowledge representation together with graph data construction methods. In addition, we categorize the feature selection process according to the availability of supervision information (class labels and pairwise constraints) and according to the evaluation criterion of performance while presenting some of the well-known state-of-the-art score-based ranking methods.

In the second chapter, we provide a survey of the most popular filter-type Relief-Based Algorithms and show their implicit margin-based core. Mainly, the original supervised Relief algorithm is explained thoroughly focusing on its strengths, limitations, and applications in different contexts as a context-aware algorithm. Also, we cover all variants, and extensions of Relief that were suggested to handle problems with noisy, incomplete, and multi-class data. The chapter is divided into four major sections each of which expresses Relief from a different possible interpretation (prob-

abilistic, comprehensible, mathematical, and statistical).

In the third chapter, we first suggest Relief-Sc and its robust version ReliefF-Sc. Their goal is effectively reducing data high dimensionality by finding a unique relevant feature subset in a closed-form. This is to be obtained in a semi-supervised context using side pairwise cannot-link constraints. Therefore, we first explain the change in the main notions of a margin (nearhit and nearmiss) from the supervised to the constrained context. We also formulate the constrained Relief-Sc under the mathematical interpretation of RBAs. In addition, we present the only other constrained margin-based algorithm (Simba-Sc) that is mainly used in our comparisons of Relief-Sc's performance. On the other side, we present our FCRSC method for redundancy elimination or minimizing redundancy with its main three building blocks: (1) sparse graph construction to represent feature similarities, (2) hierarchical clustering upon the latter, (3) combining margin maximization with the output of feature clustering which results in maximizing relevancy while minimizing redundancy. Finally, we experimentally validate the efficiency of Relief-Sc, ReliefF-Sc, and FCRSC on multiple UCI machine learning and two high dimensional gene-expression datasets in comparison with supervised, unsupervised and semi-supervised state-of-the-art filter feature selection methods.

In the fourth chapter, we present our active constraint selection and propagation methods with briefing their related work. In fact, we divide the chapter into two main parts. The first one explains our core contribution i.e. the process of selecting pairwise constraints to be used by the constrained margin-based feature selection algorithm Relief-Sc. The second one is the augmentation of supervision information by propagating these constraints called PACS. Finally, extensive experiments are applied on UCI [3] benchmark datasets and two high-dimensional gene expression ones to validate the performance of our methods in addition to showing the effect of randomly generated constraints (RCG) vs. actively selected constraints (ACS) in the process of constrained feature selection.

Finally, we highlight our contributions and conclude the thesis while pointing out our future perspectives.

Chapter 1

Score-Based Feature Selection

Contents

1.1 Introduction

When feeding high dimensional data to the "model" of traditional learning techniques without any proper preprocessing, unsatisfactory learning performance can be obtained due to a critical problem known as the curse of dimensionality [8]. The latter refers to a phenomenon by which data becomes sparser in high dimensional spaces leading to over-fitting the learning algorithm [9]. In addition, many of the features describing the original feature space in real-world applications might be irrelevant and redundant.

To clearly explain the notions of relevancy and redundancy, we use two features to present three different examples as can be seen in Figure (1.1). The first example presented in Figure (1.1a) shows the case when both Feature 1 and Feature 2 are considered irrelevant, this is due to their inability to discriminate data points of the two different classes (clusters). In the second example, presented in Figure (1.1b), Feature 1 is considered irrelevant; whereas, Feature 2 can clearly separate Class A from Class B, thus, it is considered relevant. Note that, usually a feature that best describes a dataset according to a specific relevance evaluation criterion is said to be relevant, which is in this example, the capability of class separation (classification problems). On the other side, the third example, presented in Figure (1.1c), is the case when Feature 1 and Feature 2 are redundant. In such cases, the information provided by the features is strongly correlated. Eliminating irrelevant and redundant features can decrease data dimensionality without any negative impact on the learning performance, on the contrary, it is said to enhance it.

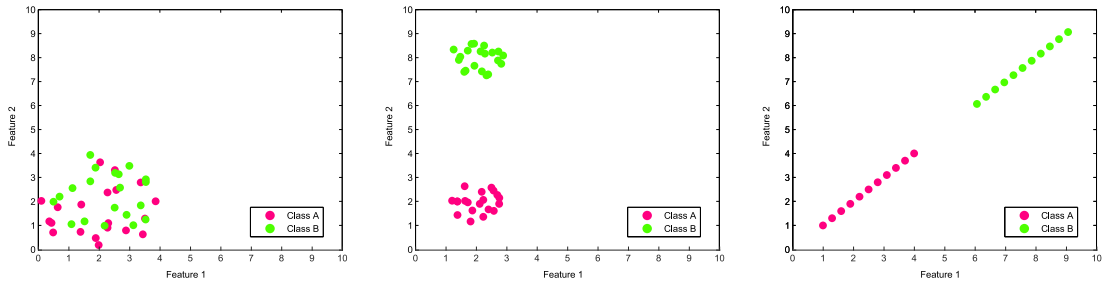
1.2 Dimensionality Reduction

It is well-known that the presence of irrelevant and redundant features can penalize the performance of a machine learning algorithm, increase its storage requirements, elevate its computational costs and make data visualization and model interpretability much harder. In order to mitigate such problems, dimensionality reduction as a data preprocessing strategy is one of the most powerful tools to be used. By its turn, it can be mainly divided into two different groups, feature extraction and feature selection [10–12].

- **Feature Extraction**: projects the initial input space onto a new one of lower dimension by combining the original features either linearly or non-linearly, thus,

changing their meaning [13, 14]. In fact, linear feature extraction is applied when the data is assumed to fall on a linear subspace or when the classes of data can be discriminated linearly, whereas, non-linear feature extraction is applied when the data pattern is assumed to be more complex and exists on a non-linear sub-manifold [15]. Some existing well-known feature extraction algorithms are Fisher Linear Discriminant Analysis (FLDA) [16], Kernel FLDA [17], Supervised Principal Component Analysis (SPCA) [18], Principal component Analysis (PCA) [19], Multidimensional Scaling (MDS) [20], Isomap [21], Local Linear Embedding (LLE) [22] and t-distributed Stochastic Neighbor Embedding (t-SNE) [23].

- **Feature Selection**: Feature selection aims at selecting the features that best describe a dataset out of a larger set of candidate features for the sake of producing a lower dimensional space without any transformation or change on the physical meaning of the original features [12, 14, 24, 25]. For classification problems, feature selection aims at selecting the highly discriminant features. In other words, it aims at selecting the features that best discriminate between data points belonging to different classes. Some well-known feature selection algorithms are Pearson Correlation Coefficient (PCC) [24], Mutual Information (MI) [15], Fast Correlation-based Filter (FCBF) [26], Sequential Forward Selection (SFS), Sequential Backward Selection (SBS) [15], and Minimum Redundancy-Maximum Relevance (mRMR) [27].
  Note that, the original input feature set is usually composed of the following four groups of features: (a) completely irrelevant, (b) weakly relevant and redundant, (c) weakly relevant but non-redundant, and (d) strongly relevant features [25].



(a) Irrelevant features (b) Relevent Feature 2 and irrelevent Feature 1 (c) Redundant features

Figure 1.1: Three examples illustrating the notions of relevancy and redundancy. (a) shows two irrelevant features; (b) shows one relevant feature (Feature 2) and one irrelevant feature (Feature 1); and (c) shows two redundant features.

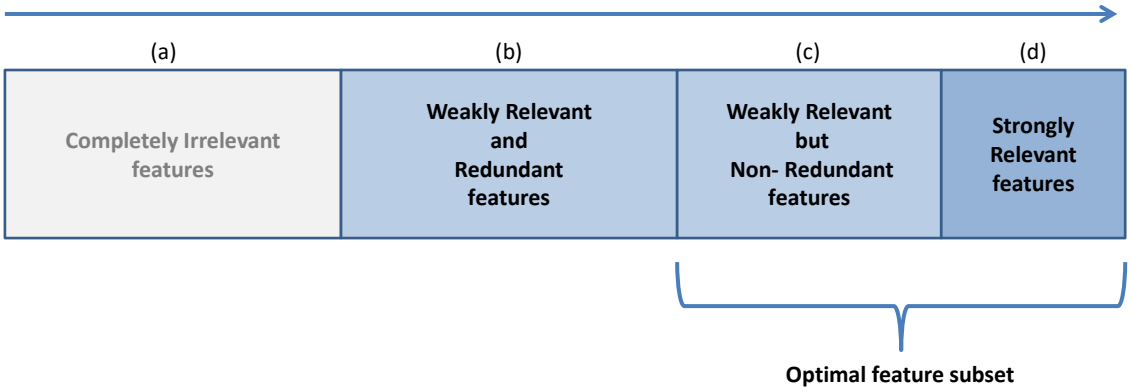


Figure 1.2: The four kinds of possible feature groups within an original feature set.

As the first two groups can significantly degrade the performance of learning algorithms and decrease their computational efficiency [28–30], it is expected from a good feature selection algorithm to be able to keep features from within groups (c) and (d). Figure (1.2) presents the four existing groups and identifies the optimal desired feature subset.

Both of the above dimensionality reduction tools, feature extraction and feature selection, are effective and capable of improving learning performance, decreasing memory storage, enhancing computational efficiency, and building better generalization models. However, feature selection costs less in terms of computation and is superior in terms of better readability and interpretability. As feature extraction maps the original feature space into a new one of a lower dimension, linking the features from the original feature space to the extracted one becomes difficult. Thus, unlike in feature selection, further analysis of the extracted features turns to be problematic and their physical meaning is not maintained. It is important to note that sometimes keeping the original meaning of features is crucial, like in genetic analysis, by which, determining which genes are responsible for a specific disease is the goal [8, 24]. Therefore, in our work, we are interested in feature selection methods.

Before moving forward with feature selection, we declare the symbols and notations that will be used in this chapter together with the main aspects of graph data representation.

1.3 Data and Graph Representation

In our work, we consider a data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ where N is the number of data points and F is the number of features. To be clear, it is possible to consider this matrix $\boldsymbol{X}$ from two perspectives:

- The data points perspective, where the data matrix $\boldsymbol{X}$ is represented as follows:

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1 \\ \cdots \\ \boldsymbol{x}_n \\ \cdots \\ \boldsymbol{x}_N \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1i} & \cdots & x_{1F} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{ni} & \cdots & x_{nF} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{N1} & \cdots & x_{Ni} & \cdots & x_{NF} \end{pmatrix} \tag{1.1}$$

  Each of the N rows of matrix $\boldsymbol{X}$ represents a data point[1] $\boldsymbol{x}_n \in \mathbb{R}^F$. Thus, $\boldsymbol{x}_{ni}$ is the value of the n-th data point over the i-th feature $\boldsymbol{A}_i$.

- The features perspective, where the data matrix $\boldsymbol{X}$ is represented as follows:

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{A}_1 & \cdots & \boldsymbol{A}_i & \cdots & \boldsymbol{A}_F \end{pmatrix} = \begin{pmatrix} A_{11} & \cdots & A_{i1} & \cdots & A_{F1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ A_{1n} & \cdots & A_{in} & \cdots & A_{Fn} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ A_{1N} & \cdots & A_{iN} & \cdots & A_{FN} \end{pmatrix} \tag{1.2}$$

  Each of the F columns of matrix $\boldsymbol{X}$ above represents a feature[2] $\boldsymbol{A}_i \in \mathbb{R}^N$. Note that, $\boldsymbol{A}_{in}$ is equivalent to $\boldsymbol{x}_{ni}$ and holds the same value. It can be also read as the value of the i-th feature over the n-th data point $\boldsymbol{x}_n$.

Throughout the report, italic letters are used to denote scalars and bold letters are used to denote vectors or matrices (e.g. x_{n1}, $\boldsymbol{x}_n$, $\boldsymbol{X}$).

1.3.1 Graph Data Representation

Liu and Zhang [31] have claimed that by constructing a graph using the data in the original feature space, some of the data's intrinsic properties are reflected. In other words, the graph structure reveals the inherent characteristics of the original data. Therefore, it is assumed that finding a smaller set of features that can best preserve the

[1] also known as an instance, example or observation.

[2] also known as a variable or attribute.

graph structure, is said to have the most informative and important features.

Generally, a very fundamental step in graph-based methods is graph construction. The latter is divided into graph adjacency determination and graph weight assignment. Regarding the graph adjacency determination, a set of well-known methods like K-nearest neighbor, ϵ-ball based, and fully connected graph are widely used [32]. On the other side, for graph weight assignment, another set of well-known methods is used, some of which are Gaussian kernel, inverse Euclidean distance, and cosine similarity [33, 34].

In addition, according to graph theory, the structure information of a graph can be obtained from its spectrum [35]. Thus, spectral graph theory [36] represents a solid theoretical framework, by which, multiple powerful existing feature selection methods depend on [35, 37–40]. Therefore, we decided to brief the corresponding basic graph aspects.

For the training set $\boldsymbol{X}$ with N points, let $G = (\mathcal{V}, E)$ be the undirected graph constructed from $\boldsymbol{X}$, where $\mathcal{V} = \{\mathcal{V}_1, \ldots, \mathcal{V}_N\}$ is the set of vertices and E is the set of edges. Each vertex $\mathcal{V}_n$ in this graph represents a data point $\boldsymbol{x}_n$ and each edge connecting two vertices $\mathcal{V}_n$ and $\mathcal{V}_m$ carries a non-negative weight $s_{nm} \geq 0$. The graph weights are represented by an NxN similarity matrix $\boldsymbol{S} = (s_{nm})_{n,m=1,\ldots,N}$ holding the pairwise similarities between all data points as follows:

$$\boldsymbol{S} = \begin{pmatrix} 0 & s_{12} & \cdots & s_{1N} \\ s_{21} & 0 & \cdots & s_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ s_{N1} & s_{N2} & \cdots & 0 \end{pmatrix} \tag{1.3}$$

G is an undirected graph, $s_{nm} = s_{mn}$, which means, the similarity matrix is symmetric. Also, note that each pairwise similarity s_{nm} is within the range $0 \leq s_{nm} \leq 1$.

According to von Luxburg [32], there are several approaches to build a graph out of a given dataset with known pairwise similarities s_{nm} between its data points. Before we get to brief the most common similarity graph-building methods, it is important to note that their main aim is modeling the local pairwise neighborhood relationships between data points.

- **ϵ-neighborhood graph**: In this type of graph, every two vertices that are associated with two data points having a distance less than a threshold ϵ, are con-

nected. In other words, a circle of radius ϵ and center $\boldsymbol{x}_n$ defines which vertices are connected to the vertex $\mathcal{V}_n$ corresponding to $\boldsymbol{x}_n$. Since the distances between connected points will all be within the same range (maximum ϵ), there is no need for weighting the edges. Instead, an edge only exists between two vertices when their corresponding data points are considered neighbors yielding an unweighted graph (the weight of an edge is either 0 or 1).

- K**-nearest neighbor graph**: In this type of graph, the goal is to connect vertex $\mathcal{V}_n$ with vertex $\mathcal{V}_m$ if $\boldsymbol{x}_m$ is among the K-nearest neighbors of $\boldsymbol{x}_n$. Since in this definition the neighborhood relationship might not be mutual (e.g $\boldsymbol{x}_n$ is in the K-nearest neighbors of $\boldsymbol{x}_m$, however, not vice versa), it outputs a directed graph.

  Two ways can be used to make this graph undirected:

  - **The normal** K**-nearest neighbor graph**: built through ignoring the directions of the edges, which means, connecting two vertices $\mathcal{V}_n$ and $\mathcal{V}_m$ with an undirected edge if $\boldsymbol{x}_n$ is among the K-nearest neighbors of $\boldsymbol{x}_m$ or $\boldsymbol{x}_m$ is among the K-nearest neighbors of $\boldsymbol{x}_n$.
  - **The mutual** K**-nearest neighbor graph**: built through connecting two vertices $\mathcal{V}_n$ and $\mathcal{V}_m$ if both of their corresponding data points are among the K-nearest neighbors of each other.

  In both cases, the edges are weighted by the similarity value between their endpoints.

- **Fully connected graph**: In this type of graph, all data points with positive similarity to each other are connected and each edge is weighted by s_{nm}. As it is expected from the graph to represent the local neighborhood relationships of data, this type of graph construction is usually only chosen if the similarity function itself models local neighborhoods.

  In fact, there are some well-known similarity functions that can be used for the construction of a graph such as:

  - **The Gaussian similarity function** evaluated as follows:

$$s_{nm} = e^{-\frac{\|x_n - x_m\|^2}{2\sigma^2}} \tag{1.4}$$

    where the parameter σ is a user-defined constant that controls the width of the neighborhood and $\|\boldsymbol{x}_n - \boldsymbol{x}_m\|$ denotes the distance between $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$.

– **The self-tuning (auto-adaptive) Gaussian similarity function** evaluated as follows:

$$s_{nm} = e^{-\frac{\|\boldsymbol{x}_n - \boldsymbol{x}_m\|^2}{2\sigma_n\sigma_m}} \tag{1.5}$$

where the local scaling parameters σ_m and σ_m are calculated by studying the local statistics of the neighborhood of points $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$ respectively. For instance $\sigma_n = \|\boldsymbol{x}_n - \boldsymbol{x}_k\|$ where K is a user-defined constant that specifies the K-th neighbor of $\boldsymbol{x}_n$ [41].

– **The inverse of Euclidean distance similarity function** calculated as follows:

$$s_{nm} = \frac{1}{\frac{\|\boldsymbol{x}_n - \boldsymbol{x}_m\|^2}{\sigma^2} + 1} \tag{1.6}$$

– **The Cosine similarity function** used to measure the similarity between two vectors by calculating the cosine of the angle between them, evaluated as follows:

$$s_{nm} = |cos(\boldsymbol{x}_n, \boldsymbol{x}_m)| = \frac{|\boldsymbol{x}_n{}^T \cdot \boldsymbol{x}_m|}{\|\boldsymbol{x}_n\| \|\boldsymbol{x}_m\|} \tag{1.7}$$

Now that we mentioned the graph-building methods, we highlight one of their important characteristics i.e. the degree d_{nn} of a vertex $\mathcal{V}_n \in \mathcal{V}$. It is equal to the sum of weights of all edges connected to this vertex. It is defined as the diagonal matrix $\boldsymbol{D} = (d_{nm})_{n,m=1\ldots N}$ calculated as follows:

$$\boldsymbol{D} = \begin{pmatrix} d_{11} & 0 & \cdots & 0 & 0 \\ 0 & d_{22} & 0 & \cdots & 0 \\ \vdots & 0 & \ddots & 0 & \vdots \\ 0 & \vdots & 0 & d_{nn} & 0 \\ 0 & 0 & \cdots & 0 & d_{NN} \end{pmatrix} \tag{1.8}$$

where,

$$d_{nn} = \sum_{m=1}^{N} s_{nm} \tag{1.9}$$

It is worth to note that the degree d_{nn} of a node $\mathcal{V}_n$ can be considered as a local density measure at $\boldsymbol{x}_n$ since d_{nn} increases with the increase of the number of data points that are close to $\boldsymbol{x}_n$ [42].

1.3.2 Graph Laplacian Matrices

Based on the definitions of the similarity matrix $\boldsymbol{S}$ and degree matrix $\boldsymbol{D}$ of the previous section, the unnormalized Laplacian matrix can be determined as follows:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{S} \tag{1.10}$$

where $\boldsymbol{S}$ and $\boldsymbol{D}$ are defined in Equations (1.3) and (1.8) respectively. The Laplacian matrix $\boldsymbol{L}$ satisfies the properties of being symmetric, positive semi-definite, and the below Equation (1.11).

For every vector $\boldsymbol{v} \in \mathbb{R}^N$:

$$\boldsymbol{v}^T \boldsymbol{L} \boldsymbol{v} = \frac{1}{2} \sum_{n,m=1}^{N} s_{nm}(v_n - v_m)^2 \tag{1.11}$$

Typically, we denote the eigenvalues of the Laplacian matrix $\boldsymbol{L}$, sorted in their increasing order, by $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_N$ and their corresponding eigenvectors $\boldsymbol{v}_1$,$\boldsymbol{v}_2$,...,$\boldsymbol{v}_N$ satisfying $\boldsymbol{L}\boldsymbol{v} = \lambda \boldsymbol{v}$.
It is important to note that the data separation information according to the graph Laplacian actually starts being available from the second eigenvector and on, this is since $\lambda_1 = 0$ and its associated vector $\boldsymbol{v}_1 = (1/\sqrt{N})\mathbf{1}$ always holds, where $\mathbf{1}$ is a vector of size N such that $\mathbf{1} = (1,1,...,1)^T$.

Note that, the diagonal elements of the similarity matrix $\boldsymbol{S}$ do not affect or change the unnormalized graph Laplacian. Thus, each similarity matrix which coincides with $\boldsymbol{S}$ on all off-diagonal positions yields the same unnormalized graph Laplacian. In particular, self-edges in a graph do not change the corresponding graph Laplacian. Therefore, these are generally set to zero ($s_{nn} = 0$).

On the other side, there are two normalized Laplacian matrices, by which, one is symmetric and the other is asymmetric (random walk) [32].

- The normalized symmetric Laplacian matrix of the graph is defined as follows:

$$\boldsymbol{L}_{sym} = \boldsymbol{D}^{-1/2} \boldsymbol{L} \boldsymbol{D}^{-1/2} = \boldsymbol{I} - \boldsymbol{D}^{-1/2} \boldsymbol{S} \boldsymbol{D}^{-1/2} \tag{1.12}$$

  where $\boldsymbol{D}^{1/2}$ is the diagonal matrix defined upon $\boldsymbol{D}_{nn}^{1/2} = \sqrt{d_{nn}}$. Thus, $\boldsymbol{D}_{nn}^{-1/2} = \frac{1}{\sqrt{d_{nn}}}$ with the assumption that $d_{nn} \neq 0$ and $\boldsymbol{L}_{sym}$ also satisfies the properties of being symmetric, positive semi-definite, and the below Equation (1.13).

For every vector $\boldsymbol{v} \in \mathbb{R}^N$:

$$\boldsymbol{v}^T \boldsymbol{L}_{sym} \boldsymbol{v} = \frac{1}{2} \sum_{n,m=1}^{N} s_{nm} (\frac{v_n}{\sqrt{d_{nn}}} - \frac{v_m}{\sqrt{d_{mm}}})^2 \tag{1.13}$$

Also, the smallest eigenvalue is given by $\lambda_1 = 0$ and its associated eigenvector $\boldsymbol{v}_1 = (\frac{1}{\sqrt{\sum_n d_{nn}}}) \boldsymbol{D}^{1/2} \mathbf{1}$ where $\mathbf{1}$ is a vector of size N such that $\mathbf{1} = (1,1,...,1)^T$.

- The normalized asymmetric Laplacian matrix, also known as random walk [32], of the graph is defined as follows:

$$\boldsymbol{L}_{rw} = \boldsymbol{D}^{-1} \boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1} \boldsymbol{S} \tag{1.14}$$

where $\boldsymbol{D}^{-1}$ is the diagonal matrix defined upon $\boldsymbol{D}^{-1}_{nn} = \frac{1}{d_{nn}}$ with the assumption that $d_{nn} \neq 0$ and the eigenvector $\boldsymbol{v}_1$ of $\boldsymbol{L}_{rw}$ is equal to $\boldsymbol{v}_1 = \frac{1}{\sqrt{N}} \mathbf{1}$ knowing that the smallest eigenvalue $\lambda_1 = 0$.

Note that, the above mentioned graphs are generally dependent on the values of the parameters ϵ, K, or σ. Without finding appropriate values of these parameters with respect to the data structure, the built similarity matrix may not be able to reflect the real similarity among data points.

Now that we have presented the symbols, notations and basic notions of data representation and graph construction, we show the categorization of feature selection into unsupervised, supervised and semi-supervised according to the integration of contextual knowledge.

1.4 Feature Selection with Contextual Knowledge

According to the availability of supervision information, feature selection is widely categorized into unsupervised when no supervision information is available at all [30, 38], as supervised when data is fully supervised [27, 43], and as semi-supervised when supervision information exists over only a few data points [10, 39].

- **Unsupervised methods**: in the context of unsupervised learning where no supervision information is available, it is difficult to define an efficient criterion for evaluating candidate features. In fact, unsupervised feature selection is said to be fabricated for clustering problems. Thus, starting with a completely unlabeled

training set, it utilizes all data points that are available in the feature selection phase for the sake of finding robust criteria to define feature relevance. Multiple unsupervised methods for selecting features were proposed in the literature. These tend to use various evaluation criteria like the ability to preserve data neighborhood graph, exploiting data similarity, maximizing feature variances, and the ability to preserve local discriminative information [24, 30, 35, 38, 44–48].

- **Supervised methods**: on the contrary to the unsupervised context, supervised feature selection methods are generally fabricated for classification and regression problems [13]. As for classification, feature selection aims at finding the features that can best discriminate data points of different classes. For example, a feature can be selected as relevant if it is highly correlated with the vector of class labels [9]. In this context, first a feature subset is selected using one of the available feature selection methods applied on the training data [27, 43, 49–51]. Afterward, the learning algorithm is trained over the selected subset of features. Then, the built model is applied to unseen data (an unlabeled testing set described by the same set of features) for predicting class labels. Note that, the cost of labeling data points by domain human experts is very high in terms of time and effort and may not always be error-free (some data might be labeled falsely [52]).

- **Semi-supervised methods**: in brief, while traditional supervised feature selection methods work when the data is fully labeled, unsupervised ones work without any supervision information. However, in many real-world applications, the amount of supervision information (e.g. the number of labeled data) might be limited providing insufficient supervision information to supervised feature selection methods. Similarly, unsupervised feature selection methods can be effective using unlabeled data only, however, without the ability to benefit from the available supervision information. Hence, it is preferable to use semi-supervised methods that can evaluate the relevance of features taking into account both labeled and unlabeled data [10, 53–56].

Therefore, in a training set $\boldsymbol{X}$ of N data points we may have two subsets depending on the label availability: $\boldsymbol{X}_l = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_l\}_{l \neq 0}$ with the class label corresponding to each data point and $\boldsymbol{X}_u = \{\boldsymbol{x}_{l+1}, \boldsymbol{x}_{l+2}, \ldots, \boldsymbol{x}_{l+u}\}_{u \neq 0}$, which are unlabeled. When $l = 0$, all data points are unlabeled and the learning context is said to be unsupervised. When $u = 0$, all data points are labeled and the learning context is said to be supervised. However, in real world problems, where we typically have only few labeled and many

unlabeled data points ($l << u$) the context is said to be semi-supervised. Note that, $N = l + u$.

1.4.1 Types of Supervision Information

Generally, class labels are the first to come to mind when mentioning supervision information. These specify which class each data point should belong to. Accordingly, supervised and semi-supervised feature selection methods are usually explained in terms of the available amount of class labels. However, in real-world applications, there exists a cheaper kind of supervision information i.e. *pairwise constraints*. On the contrary to class labels, these constraints only specify whether a pair of data points should belong to the same group (must-link constraint) or different groups (cannot-link constraint) without identifying the groups themselves.

1.4.1.1 Class Labels

As can be seen in Figure (1.3), in supervised or semi-supervised contexts, feature selection utilizes fully or partially labeled data respectively. In both cases, it aims at evaluating the relationship between the features and their provided class label information. Thus, considering that in the training set $\boldsymbol{X}$ each data point $\boldsymbol{x}_n$ is associated with a class label y_n, we denote by $\boldsymbol{y}_l$ the vector of labels defined as follows:

$$\boldsymbol{y}_l = \begin{pmatrix} y_1 \\ ... \\ y_n \\ ... \\ y_N \end{pmatrix} \tag{1.15}$$

where $y_n \in \{1,..,c,...,C\}$ and C is the number of classes of the data. Having C classes, we denote by N_c the number of data points in each class c.

1.4.1.2 Pairwise Constraints

Pairwise constraints are a cheaper kind of prior knowledge. They guide learning algorithms and allow multiple unsupervised ones to improve their performance [57]. It is easier for a user to specify whether some pairs of data points belong to the same class or not, in other words, similar or dissimilar, than it is to specify class belongings. In fact, class labels can be directly transformed into must-link and cannot-link constraints, however, constraints cannot be transformed into class labels. This is straight-

forward since the amount of information provided by class labels is superior. Hence, one way of building the must-link and cannot-link constraint sets is directly from the class labels by connecting the data points that share the same label with a must-link and connecting the data points that have different labels with a cannot-link [53, 58, 59]. Another way is to find these sets by directly adding them to data points using Active Learning as will be suggested and discussed in chapter 4.

The sets of must-link and cannot-link constraints consisting of pairs of data points from the training set $\boldsymbol{X}$, denoted by ML and CL respectively, are defined as follows:

- $ML = \{(\boldsymbol{x}_n, \boldsymbol{x}_m) \mid \quad \boldsymbol{x}_n \text{ and } \boldsymbol{x}_m \text{ belong to the same group}\}$
- $CL = \{(\boldsymbol{x}_n, \boldsymbol{x}_m) \mid \quad \boldsymbol{x}_n \text{ and } \boldsymbol{x}_m \text{ belong to different groups}\}$

Pairwise constraints are used to evaluate the relevance of each feature according to its constraint preserving ability. In the context of spectral theory, two graphs G^{ML} and G^{CL} can be constructed to represent them by using the data points of ML and CL respectively. Accordingly, an edge is created between two nodes of the graph G^{ML} (or G^{CL}) when their corresponding data points are must-linked (or cannot-linked). Thus, the similarity matrices holding the edge weights between every two nodes in G^{ML} and G^{CL} are defined as follows:

$$s_{nm}^{ML} = \begin{cases} 1 & \text{if } (\boldsymbol{x}_n, \boldsymbol{x}_m) \in ML \\ 0 & \text{otherwise} \end{cases} \tag{1.16}$$

$$s_{nm}^{CL} = \begin{cases} 1 & \text{if } (\boldsymbol{x}_n, \boldsymbol{x}_m) \in CL \\ 0 & \text{otherwise} \end{cases} \tag{1.17}$$

In the three mentioned learning contexts: supervised, unsupervised, and semi-supervised, the process of feature selection follows a general workflow. The latter is illustrated in Figure (1.3) and will be detailed in the following section.

1.5 General Procedure of Feature Selection

By following the flow chart of the general procedure of feature selection presented in Figure (1.3), there are typically four basic steps [12]. These are known as subset generation, subset evaluation, stopping criterion and result validation [11].

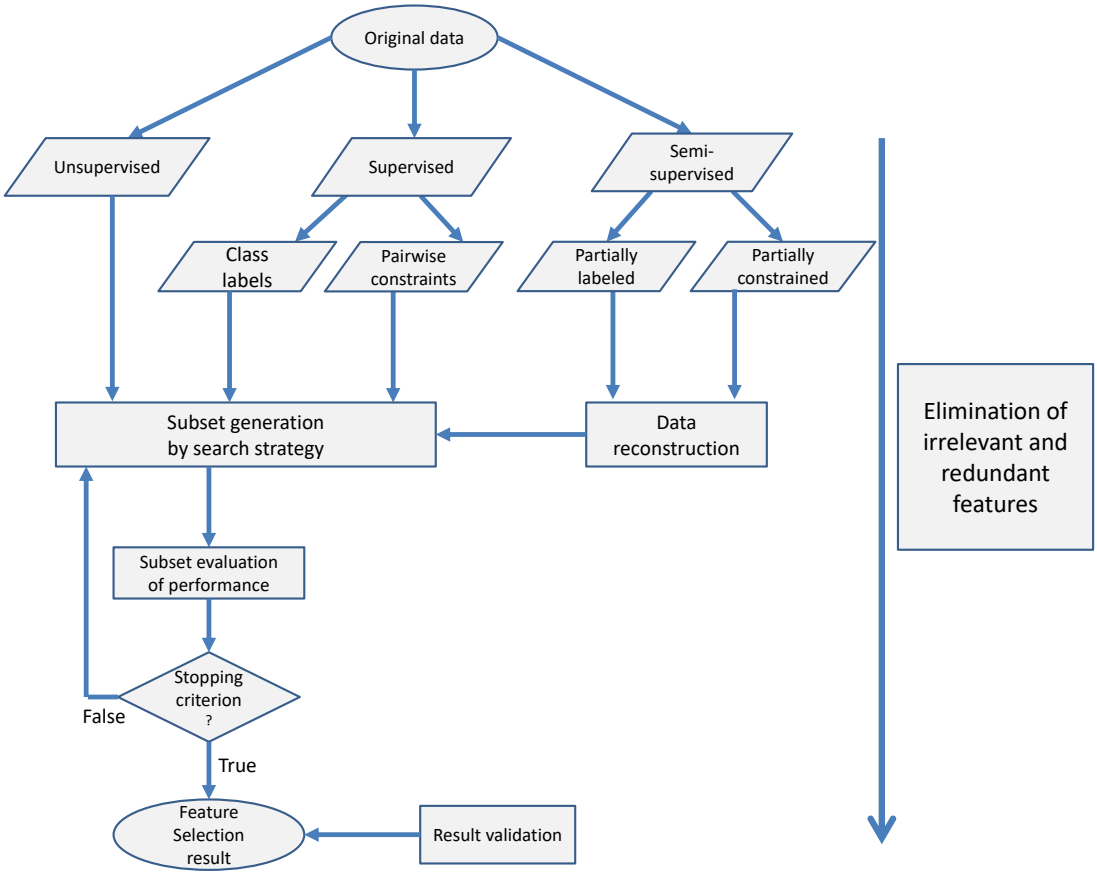


Figure 1.3: A flow chart showing the general procedure of feature selection while taking the contextual knowledge into consideration.

1.5.1 Subset Generation

Initially, subset generation, i.e. a search procedure, utilizes specific search strategies to produce candidate feature subsets ready for the following evaluation step. It is based on complete, sequential and random search strategies. (1) The complete search strategy guarantees finding the optimal subset with respect to the used evaluation criterion. One example of complete search is the exhaustive search where no optimal feature subset is missed; However, this might induce high computational cost (i.e. $O(2^F)$ when the number of features F is large). (2) The sequential search strategy is computationally less expensive; However, it might not find the optimal set. Some approaches to this type of search are Sequential Forward Selection and Sequential Backward Selection. The former starts with an empty set and adds one feature at a time according to the evaluation criterion, whereas, the latter starts with the full set of features and removes one feature at a time. (3) Random search strategy escapes the chance of falling into a local optima by starting with a randomly chosen set of features and then contin-

uing either with sequential search or with simply other random feature sets.

1.5.2 Evaluation Criterion of Performance

At this step, each of the candidate feature subsets is examined with respect to its preceding best subset by a particular evaluation criterion. If the evaluation's output confirms that the new subset is better than the one before, the former replaces the latter. As can be seen in Figure (1.4), feature selection can be categorized into three main methods according to the evaluation criterion of performance. These methods are: filter methods [38, 40], wrapper methods [60, 61] and embedded methods [62, 63]. Filter methods are also known as independent methods since they are independent of any learning algorithm, while, wrapper and embedded methods are known as dependent ones since they utilize the performance of a learning algorithm in the process of selecting features [11, 12].

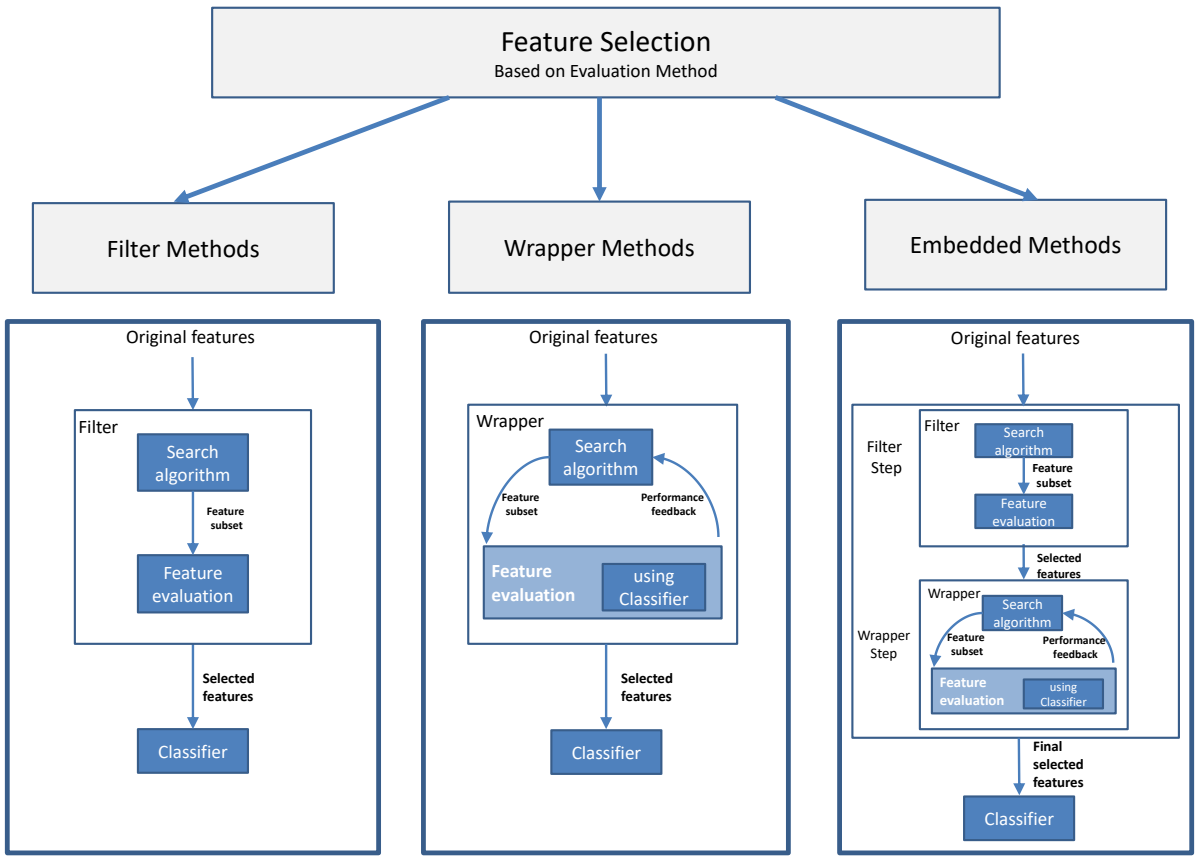


Figure 1.4: The broad categorization of feature selection into filter, wrapper and embedded methods based on the evaluation criterion.

1.5.2.1 Filter methods

Filter methods measure the goodness of generated feature subsets by relying on the intrinsic characteristics of the data training set independently from any learning algorithm. Hence, although filter methods are said to be computationally efficient, their selected feature subsets might not be optimal for the used learning algorithm. This is caused by the fact that no specific learning algorithm is guiding the process of feature selection. Generally, any filter method works in two consecutive steps. Step 1, consists of ranking the features in the order of their importance with respect to a specific evaluation criterion. This criterion is further divided into univariate [38, 64] and multivariate schemes [27, 65]. In the former, each feature is evaluated individually and independently from other features, whereas, in the latter, multiple features are evaluated and ranked together in a batch mode. Step 2 consists of filtering out the features of lowest ranks, thus, obtaining the final selected subset.

Some of the well-known filter evaluation criteria are:

- Measures of features discriminative ability to separate classes (also known as distance, divergence and separability measures) [37, 66, 67].
- Measures of features correlation with class labels (also known as dependency measures) [24, 68].
- Information based measures that typically evaluate the information gain from a feature [27, 69, 70].
- Measures of features manifold structure preserving ability [35, 38, 71].
- Measures of features ability in reconstructing the original data [45, 72].

1.5.2.2 Wrapper methods

Wrapper methods require a predefined learning algorithm, by which, the process of feature selection is wrapped around. In fact, wrapper methods utilize the performance of this predetermined learning algorithm to evaluate the importance of features. Generally, superior performance and optimal feature subsets are guaranteed with such methods as they focus on choosing the features that best improve the predictive accuracy of a specific learning algorithm, however, they also tend to be computationally more expensive. To sum up, after deciding on the learning algorithm, a typical wrapper method performs two steps. In step 1, it searches for a subset of features. In step

2, it evaluates the selected subset using the chosen learning algorithm. Steps 1 and 2 are repeated until the stopping criterion (section 1.5.3) is reached [60, 73].

1.5.2.3 Embedded methods

For the sake of a more efficient feature selection, embedded methods are a trade-off between both filter and wrapper methods. By which, they generally try to find a feature subset that maintains good performance of learning algorithms while keeping computational cost in an acceptable range. For instance, one may consider a filter approach to remove low ranked features in terms of relevance or importance, and then apply a wrapper method on the selected subset to find the best features from within [46, 65, 74].

Out of these three widely used feature selection approaches, we are interested in filter methods as they are considered simple, fast and efficient. Moreover, they can be applied as an independent preprocessing step before any mining algorithm.

1.5.3 Stopping Criterion

The stopping criterion decides when the process of feature selection should be terminated. In fact, both the generation and evaluation steps are repeated until the stopping criterion is met. The latter can be determined by:

- The subset generation step, where the process can be stopped either when a specific number of selected features or a maximum number of iterations is reached.
- The evaluation step, where feature selection termination can be determined by the stability of performance, which means, adding or removing features would not impose any performance change or improvement. It can also be determined by obtaining a sufficiently good feature subset, in other words, a subset that obtains an evaluation measure, i.e. greater than an acceptance threshold.

1.5.4 Result Validation

Finally, the selected feature subset (i.e. considered the best compared to all other generated subsets) has to be validated. Accordingly, result validation can be done by simply evaluating the selected subset using domain prior knowledge. For example, by monitoring the change in the classification performance of a classifier with respect to the change in the feature subsets. For instance, if the classification accuracy rate

records a greater value using the selected subset than using the original feature space, the result is considered good.

Note that, data classification is the problem of automatically determining to which category, out of a given set of categories, a new data point belongs. This is done through a process of two consecutive phases. (1) Building a model using a data training set, by which, the category or class membership of each data point is a priori known. (2) Testing the built model using a data testing set, by which, the category or class membership of each data point is unknown. An example of a classification problem is automatically deciding whether a person is healthy or diseased based on his observed medical characteristics (features like: age, gender, family history, blood pressure, *etc*). In fact, some well-known classification algorithms usually used for this type of validation are as follows:

- *K*-Nearest Neighbor (*K*-NN) [75]: a simple non-parametric method that can achieve high performance when the number of data points is sufficiently big. It utilizes only the spatial distributions of empirical samples without any previous assumptions about their class distributions, where a new data point is classified by the class of the majority of its *K*-nearest points.

- Support Vector Machines (SVM) [76]: a set of well-known general learning methods that have become very popular in the last few decades. A SVM classifier maps data points into another space such that, a gap (called sample-margin) between the mapped points of different classes is maximized. Accordingly, new data points are mapped into that same space and classified based on which side of the gap they fall. Basically, multi-class SVM can be done by dividing a multi-class problem into a set of two-class ones (e.g. one-against-one and one-against-all methods). Besides, different kernel functions like linear, sigmoid, polynomial and radial basis function (RBF) can be used.

- Naive Bayes (NB) [77]: a probabilistic classifier based on Bayes theorem. It applies classification with a naive (strong) independence assumption among features. In other words, it considers that given the class labels, features are conditionally independent of each other.

- Decision Tree (C4.5) [78]: a well-known classifier that applies an entropy-based criterion on a set of training data to build the decision tree. For instance, the data points can be split into smaller subsets by using a feature as the decision rule. For this purpose, the algorithm measures the *information gain* at each split.

1.5.4.1 Relevancy Performance Evaluation Metrics

There exist multiple supervised performance evaluation metrics that are related to data classification and can be obtained after applying a classifier on the selected set of features. Three widely used metrics are accuracy, precision, and recall. To clearly explain them, we consider having two classes, a positive class and a negative class to build the confusion matrix. It holds the four different combinations of predicted and actual values of these two classes as can be seen in Figure (1.5).

1. True Positives (*TP*): the total number of correct predictions that are positive, which means, were predicted to be positive knowing that they actually belong to the positive class.

2. False Positives (*FP*): the total number of incorrect predictions that are positive, which means, were predicted to be positive knowing that they actually belong to the negative class.

3. True Negative (*TN*): the total number of correct predictions that are negative, which means, were predicted to be negative knowing that they actually belong to the negative class.

4. False Negative (*FN*): the total number of incorrect predictions that are negative, which means, were predicted to be negative knowing that they really belong to the positive class.

	Actual Positive	**Actual Negative**
Predicted Positive	***TP (True Positive)***	***FP (False Positive)***
Predicted Negative	***FN (False Negative)***	***TN (True Negative)***

Figure 1.5: The confusion matrix, a table of the four different combinations of predicted (rows) and actual (columns) class values. It is very useful for understanding and measuring the accuracy, precision, and recall.

- Classification Accuracy: a well-known evaluation metric defined as a percentage of correct predictions. Thus, it evaluates how many data points were correctly

classified using the selected features by classifiers like the ones presented in 1.5.4.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.18}$$

- Precision and Recall: another two well-known evaluation metrics usually used together and can be applied in the context of classification. Precision answers the question of what proportion of positive predictions was actually correct. Whereas, recall, also called sensitivity, answers the question of what proportion of actual positives was identified correctly.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1.19}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{1.20}$$

1.6 Ranking Feature Selection Methods based on Scores

In feature selection, filter methods can be broadly categorized as relying on subset evaluation or individual evaluation. Subset evaluation depends on some search strategy to evaluate the relevance of candidate feature subsets. On the other side, individual evaluation, also known as feature ranking/weighting, aims at sorting the features in either the increasing or decreasing order of a particular weight or score that is assigned to each of them [2]. For that, unlike subset evaluation, it assesses the degree of relevance of each feature individually.

As our work is focused on feature ranking methods, in this section, we briefly review eight of the well-known ranking methods that are based on scores and will be used throughout the upcoming chapters. Later on, also a detailed and concise review of a filter-type feature ranking family that is based on feature weights is presented in chapter 2.

The eight well-known score-based ranking methods that are briefly reviewed in this section are divided into unsupervised, supervised and semi-supervised as presented in Figure (1.6).

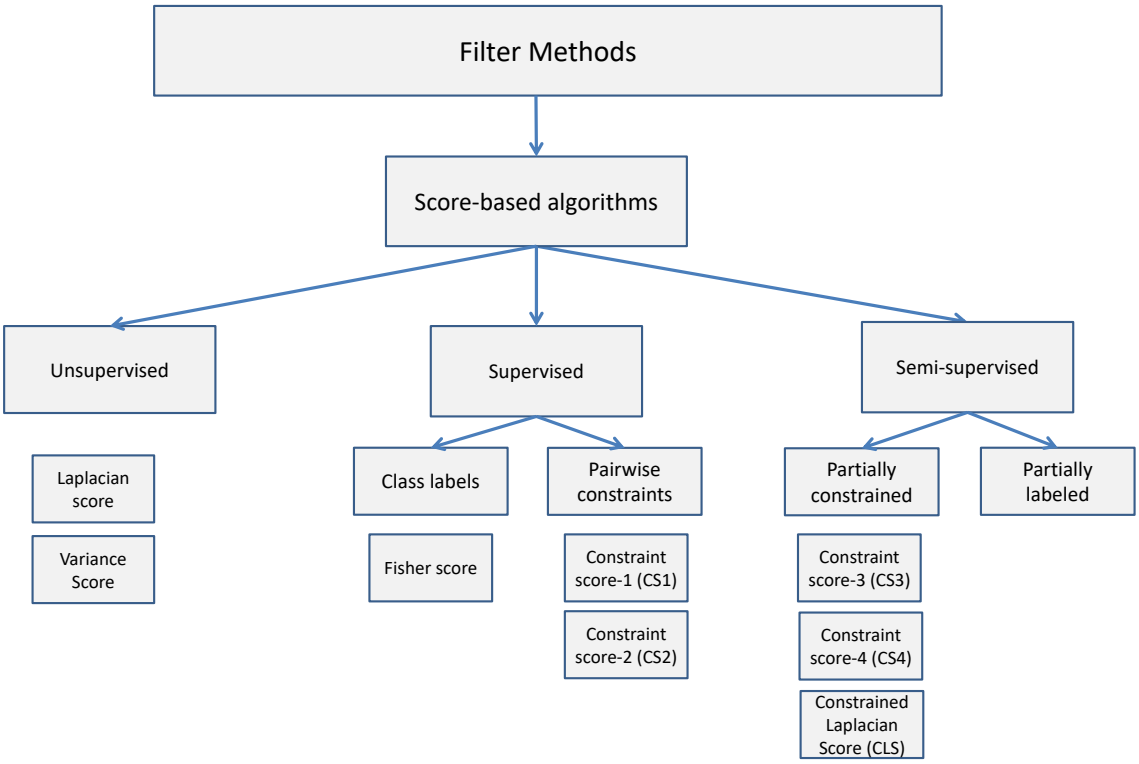


Figure 1.6: Score-based feature selection methods categorized according to the different learning contexts.

1.6.1 Unsupervised Scores

- Variance Score [64]: generally known as the simplest unsupervised feature evaluation method. It uses the variance along a specific feature to reflect its representative power. Hence, the features with the maximum variance are selected assuming that a feature with higher variance contains more information and is more relevant. This means that the features are ranked in the decreasing order of their assigned Variance scores V_i.
  Variance score depends on the following equation to evaluate features:

$$V_i = \frac{1}{N}\sum_{n=1}^{N}(A_{in} - \mu_{\boldsymbol{A}_i})^2 \tag{1.21}$$

  where N is the number of data points, A_{in} is the value of feature $\boldsymbol{A}_i$ on a data point $\boldsymbol{x}_n$ and $\mu_{\boldsymbol{A}_i} = \frac{1}{N}\sum_{n=1}^{N} A_{in}$.
  Variance score can be also interpreted as a global graph preserving method as follows [31]:

$$V_i = \frac{1}{N}\boldsymbol{A}_i^T\left(\boldsymbol{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T\right)\boldsymbol{A}_i \tag{1.22}$$

where $\boldsymbol{I}$ is the identity matrix and $\mathbf{1} \in \mathbb{R}^N$ is a vector of all ones.

- Laplacian Score [38]: a well-known unsupervised feature selection method which does not only depend on selecting the features of larger variances and higher representative power but also considers their locality preserving ability. Its key assumption is that data points within the same class should be close to each other and far otherwise. In other words, a feature is said to be "good" when two near data points in the original space are also near to each other on this particular feature, which means, it has the ability to preserve the local geometrical structure of the data. Note that, the smaller Laplacian score is the better, i.e., the features are ranked in the increasing order of their assigned Laplacian scores LS_i.
  The Laplacian score is calculated according to the following equation:

$$LS_i = \frac{\sum_{\boldsymbol{x}_n, \boldsymbol{x}_m \in \boldsymbol{X}} (A_{in} - A_{im})^2 s_{nm}}{\sum_{\boldsymbol{x}_n \in \boldsymbol{X}} (A_{in} - \mu_{\boldsymbol{A}_i})^2 d_{nn}} \tag{1.23}$$

  where d_{nn} is defined in Equation (1.9) and $\boldsymbol{D}$ is a diagonal matrix holding d_{nn} in its diagonal defined in Equation (1.8). $\boldsymbol{S}$ is neighborhood matrix between data points expressed as follows:

$$s_{nm} = \begin{cases} e^{-\frac{\|\mathbf{x}_n - \mathbf{x}_m\|^2}{\sigma^2}} & \text{if } \boldsymbol{x}_n \textit{ and } \boldsymbol{x}_m \textit{ are neighbors} \\ 0 & \text{otherwise} \end{cases} \tag{1.24}$$

  where σ^2 is a user-defined constant and "$\boldsymbol{x}_n$ *and* $\boldsymbol{x}_m$ *are neighbors*" means that either $\boldsymbol{x}_n$ is in the K-nearest neighbors of $\boldsymbol{x}_m$ or $\boldsymbol{x}_m$ is among the K-nearest neighbors of $\boldsymbol{x}_n$.
  In addition, Equation (1.23) can be represented under the framework of the spectral graph theory as follows:

$$LS_i = \frac{\tilde{\boldsymbol{A}}_i^T \boldsymbol{L} \tilde{\boldsymbol{A}}_i}{\tilde{\boldsymbol{A}}_i^T \boldsymbol{D} \tilde{\boldsymbol{A}}_i} \tag{1.25}$$

  where $\boldsymbol{L}$ is defined as in Equation (1.10) and $\tilde{\boldsymbol{A}}_i$ is defined as:

$$\tilde{\boldsymbol{A}}_i = \boldsymbol{A}_i - \frac{\boldsymbol{A}_i^T \mathbf{D1}}{\mathbf{1}^T \mathbf{D1}} \mathbf{1} \tag{1.26}$$

  and $\mathbf{1} \in \mathbb{R}^N$ is a vector of all ones.

1.6.2 Supervised Scores

1.6.2.1 Using Class Labels

- Fisher Score [64]: a well-known supervised feature selection method that seeks features with best discriminant ability. It is based on maximizing the distances between data points of different classes and minimizing the distances among points of the same class. To rank the features in the order of their relevancy, they are sorted in the decreasing order of their obtained fisher score F_i. Thus, as the value of an assigned score to a feature increases, its importance also increases.
  The Fisher score is calculated according to the following equation:

$$F_i = \frac{\sum_{c=1}^{C} N_c(\mu_{\boldsymbol{A}_i}^c - \mu_{\boldsymbol{A}_i})^2}{\sum_{c=1}^{C} N_c(\sigma_{\boldsymbol{A}_i}^c)^2} \tag{1.27}$$

  where, N_c is the number of data points in class c, $\mu_{\boldsymbol{A}_i}$ denotes the mean of the i-th feature over all data points, and $\mu_{\boldsymbol{A}_i}^c$ and $(\sigma_{\boldsymbol{A}_i}^c)^2$ are the mean and the variance of class c upon the i-th feature $\boldsymbol{A}_i$ respectively.
  Also, it is possible to compute Fisher score using the global graph-preserving method [31]. This can be done by writing Equation (1.27) as follows:

$$F_i = \frac{\boldsymbol{A}_i^T(\boldsymbol{S}_w - \boldsymbol{S}_b)\boldsymbol{A}_i}{\boldsymbol{A}_i^T(\boldsymbol{I} - \boldsymbol{S}_w)\boldsymbol{A}_i} \tag{1.28}$$

  where $\boldsymbol{S}_w = \sum_{c=1}^{C} \frac{1}{N_c}\boldsymbol{e}_c{\boldsymbol{e}_c}^T$ is the summation of the weight matrices of C within-class graphs knowing that $\boldsymbol{e}_c$ is an F-dimensional class indicator vector that holds $\boldsymbol{e}_c(n) = 1$ if $\boldsymbol{x}_n$ is in class c or 0 otherwise. In each of these within-class graphs, all data points are connected with an equal weight $1/N_c$. $\boldsymbol{S}_b = \frac{1}{N}\boldsymbol{e}\boldsymbol{e}^T$, on the other side, is the weight matrix of between-class graphs, by which, each edge connecting different classes is assigned a weight of $1/N$. Note that, $\boldsymbol{e}$ is an F-dimensional between-class indicator vector.

1.6.2.2 Using Pairwise Constraints

- Constraint Scores [40]: two supervised constrained feature selection methods that utilize pairwise constraints (must-links and cannot-links) only to evaluate the relevance of features. Their key idea is to select features having the highest constraint preserving ability. This means that a "good" feature is the one that correctly reflects that two data points are close to each other when they are connected by a must-link constraint and that they are far from one another when

they are connected by a cannot-link constraint.
The two introduced score functions are: Constraint Score-1 (CS1) and Constraint Score-2 (CS2). Both score functions are to be minimized, which means, a lower score corresponds to a more relevant feature.

CS1 is calculated according to the following equation:

$$CS1_i = \frac{\sum_{(\boldsymbol{x}_n,\boldsymbol{x}_m)\in ML}(\boldsymbol{A}_{in} - \boldsymbol{A}_{im})^2}{\sum_{(\mathbf{x}_n,\mathbf{x}_m)\in CL}(\boldsymbol{A}_{in} - \boldsymbol{A}_{im})^2} \tag{1.29}$$

CS2, a variant of CS1, is calculated according to the following equation:

$$CS2_i = \sum_{(\boldsymbol{x}_n,\boldsymbol{x}_m)\in ML}(\boldsymbol{A}_{in} - \boldsymbol{A}_{im})^2 - \lambda \sum_{(\mathbf{x}_n,\mathbf{x}_m)\in CL}(\boldsymbol{A}_{in} - \boldsymbol{A}_{im})^2 \tag{1.30}$$

Where ML and CL are the sets of must-link and cannot-link constraints defined in section 1.4.1.2 and λ is a regularization coefficient used to balance the contribution of the first and second terms of Equation (1.30). This is since the distance between points belonging to different classes is usually larger than the distance between points of the same class, thus, λ needs to be relatively small. The default value of λ, set by the authors, is 0.1 [40].

Note that, CS1 and CS2 can also be expressed under the framework of spectral theory according to the following equations:

$$CS1_i = \frac{\boldsymbol{A}_i^T \boldsymbol{L}^{ML} \boldsymbol{A}_i}{\boldsymbol{A}_i^T \boldsymbol{L}^{CL} \boldsymbol{A}_i} \tag{1.31}$$

$$CS2_i = \boldsymbol{A}_i^T \boldsymbol{L}^{ML} \boldsymbol{A}_i - \lambda \boldsymbol{A}_i^T \boldsymbol{L}^{CL} \boldsymbol{A}_i \tag{1.32}$$

where, the matrices $\boldsymbol{L}^{ML}$ and $\boldsymbol{L}^{CL}$ define the constraint Laplacian matrices calculated as: $\boldsymbol{L}^{ML} = \boldsymbol{D}^{ML} - \boldsymbol{S}^{ML}$ and $\boldsymbol{L}^{CL} = \boldsymbol{D}^{CL} - \boldsymbol{S}^{CL}$ where $\boldsymbol{D}^{ML}$ and $\boldsymbol{D}^{CL}$ are the degree matrices defined by $d_{nn}^{ML} = \sum_{m=1}^{N} s_{nm}^{ML}$ and $d_{nn}^{CL} = \sum_{m=1}^{N} s_{nm}^{CL}$ having the similarity matrices $\boldsymbol{S}^{ML}$ and $\boldsymbol{S}^{CL}$ defined by Equations (1.16) and (1.17) respectively.

1.6.3 Semi-supervised Scores with Pairwise Constraints

- Constraint Score-3 (CS3) [79]: a semi-supervised feature selection method that is fundamentally based on spectral graph theory and manifold learning. It aims

at selecting the most locality sensitive discriminant features. For that, it tries to find both the local geometrical structure and the discriminant structure of data by utilizing unlabeled data and pairwise constraints respectively. In this method, a nearest neighbor graph[3] (G^{kn}) is built using the set of must-link constraints ML and the data set $\boldsymbol{X}$. Another graph[4] (G^{CL}) is built as defined in section 1.4.1.2. The edges of these two graphs are weighted using their corresponding similarity matrices $\boldsymbol{S}^{kn}$ and $\boldsymbol{S}^{CL}$ respectively. This score is to be minimized by which a lower score means a more relevant feature.
CS3 can be calculated as follows:

$$CS3_i = \frac{\sum_{(x_n,x_m)\in X}(\boldsymbol{A}_{in}-\boldsymbol{A}_{im})^2 s^{kn}_{nm}}{\sum_{(x_n,x_m)\in X}(\boldsymbol{A}_{in}-\boldsymbol{A}_{im})^2 s^{CL}_{nm}} \tag{1.33}$$

where,

$$s^{kn}_{nm} = \begin{cases} \gamma & \text{if } \boldsymbol{x}_n \textit{ and } \boldsymbol{x}_m \in ML \\ 1 & \text{if } \boldsymbol{x}_n \textit{ and } \boldsymbol{x}_m \textit{ are unlabeled} \\ & \textit{but } \boldsymbol{x}_n \in KNN(\boldsymbol{x}_m) \textit{ or } \boldsymbol{x}_m \in KNN(\boldsymbol{x}_n) \\ 0 & \text{otherwise} \end{cases} \tag{1.34}$$

Note that, $KNN(\boldsymbol{x}_n)$ denotes the set of the K-nearest neighbors to $\boldsymbol{x}_n$. γ and k were set empirically to the values of 100 and 5 respectively [79].

As a graph-based method, the equation of CS3 can be also given by:

$$CS3_i = \frac{\boldsymbol{A}_i^T \boldsymbol{L}^{kn} \boldsymbol{A}_i}{\boldsymbol{A}_i^T \boldsymbol{L}^{CL} \boldsymbol{A}_i} \tag{1.35}$$

where $\boldsymbol{L}^{kn} = \boldsymbol{D}^{kn} - \boldsymbol{S}^{kn}$ defines the Laplacian matrix of graph G^{kn}, and the degree matrix$\boldsymbol{D}^{kn}$ is defined by $d^{kn}_{nn} = \sum_{m=1}^{N} s^{kn}_{nm}$.
Due to the large value of γ, although CS3 is semi-supervised, it is considered very similar to CS2 that neglects unlabeled data points [53].

- Constraint Score-4 (CS4) [53]: a semi-supervised constrained feature selection method that not only utilizes pairwise constraints to evaluate the relevance of features but also uses the intrinsic idea of the unsupervised Laplacian Score to benefit from the available unlabeled data. Consequently, this score is considered less sensitive to the user-defined constraint sets. It applies a simple multiplica-

[3]Also known as the within-class graph.
[4]known as the between-class graph.

tion of the unsupervised Laplacian Score from Equation (1.23) with the supervised CS1 from Equation (1.30). This score is to be minimized by which a lower score means a more relevant feature. CS4 is calculated according to the following equation:

$$CS4_i = \frac{\sum_{x_n,x_m \in \boldsymbol{X}} (A_{in} - A_{im})^2 s_{nm}}{\sum_{x_n \in \boldsymbol{X}} (A_{in} - \mu_{A_i})^2 d_{nn}} \cdot \frac{\sum_{(x_n,x_m) \in ML} (A_{in} - A_{im})^2}{\sum_{(x_n,x_m) \in CL} (A_{in} - A_{im})^2} \tag{1.36}$$

Note that, CS4 can also be expressed under the framework of spectral theory as it is also a graph-based method. The equation is given by:

$$\begin{aligned} CS4_i &= \frac{\tilde{\boldsymbol{A}_i}^T \boldsymbol{L} \tilde{\boldsymbol{A}_i}}{\tilde{\boldsymbol{A}_i}^T \boldsymbol{D} \tilde{\boldsymbol{A}_i}} \cdot \frac{\boldsymbol{A}_i^T \boldsymbol{L}^{ML} \boldsymbol{A}_i}{\boldsymbol{A}_i^T \boldsymbol{L}^{CL} \boldsymbol{A}_i} \\ &= LS_i \,.\, CS1_i \end{aligned} \tag{1.37}$$

- Constrained Laplacian Score (CLS) [54]: a semi-supervised feature selection method that evaluates the relevance of a feature according to both of its locality and constraints preserving abilities. It aims at selecting the most discriminative and informative features for data analysis and initially originates from constraining the known Laplacian score. This score considers that a relevant feature should be the one, by which, two data points that are neighbors or connected by a must-link constraint, are close to each other. In addition, it should be the feature of high variance, by which, two cannot-linked data points are well separated. This score is to be minimized. It is calculated according to the following equation:

$$CLS_i = \frac{\sum_{x_n,x_m \in \boldsymbol{X}} (A_{in} - A_{im})^2 s_{nm}^{kn}}{\sum_n \sum_{m | \exists l, (x_l,x_m) \in CL} (A_{in} - \alpha_{A_{inm}})^2 d_{nn}} \tag{1.38}$$

where,

$$s_{nm}^{kn} = \begin{cases} e^{-\frac{\|x_n - x_m\|^2}{\sigma^2}} & \text{if } \boldsymbol{x}_n \textit{ and } \boldsymbol{x}_m \textit{ are neighbors or } (\boldsymbol{x}_n, \boldsymbol{x}_m) \in ML \\ 0 & \text{otherwise} \end{cases} \tag{1.39}$$

and

$$\alpha_{\boldsymbol{A}_{inm}} = \begin{cases} \boldsymbol{A}_{im} & \text{if } (\boldsymbol{x}_n, \boldsymbol{x}_m) \in CL \\ \mu_{\boldsymbol{A}_i} & \text{otherwise} \end{cases} \tag{1.40}$$

Hindawi [59] also derived CLS in the spectral graph-based representation as follows:

$$CLS_i = \frac{\boldsymbol{A}_i^T \boldsymbol{L}^{kn} \boldsymbol{A}_i}{\boldsymbol{A}_i^T \boldsymbol{L}^{CL} \boldsymbol{D}^{kn} \boldsymbol{A}_i} \tag{1.41}$$

1.7 Feature Redundancy Analysis

As detailed in section 1.6, a common practice of score-based filter-type feature selection methods is to simply rank the features in the order of their assigned scores at the end of the individual evaluation process [59]. Then, a subset of the top-ranked features, e.g. a group of the first 10 features if the desired subset size is 10, is considered as the final selected feature subset [80]. One drawback of this straightforward ranking approach is that the selected subset might hold features that are correlated to one another. This is also known as feature redundancy. In fact, redundant features may increase dimensionality unnecessarily [81] and degrade learning performance when facing a shortage of data [82]. Therefore, multiple pieces of research suggested enhancing the representative power of the feature subset by demanding that the selected features are maximally dissimilar to each other [27, 54, 80, 83–85].

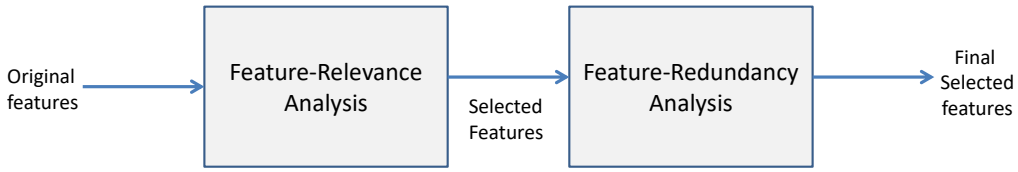


Figure 1.7: The general two-step framework of classical feature selection methods that handle maximizing the relevancy of features with respect to a specific problem while minimizing their redundancy with respect to each other.

Classical methods remove redundancy among features by generally depending on Euclidean distance, Pearson correlation and information measures [25]. Figure (1.7) shows the general procedure of such methods. For instance, one widely used method is the "Minimum Redundancy-Maximum Relevance" called (mRMR) [27]. It is a supervised multivariate feature selection method that is said to output a feature subset having the most diverse features (as non-correlated to each other as possible) while still having a high correlation with the class label.

For that, mRMR method uses two criteria that are based on mutual information.

- A maximal relevance criterion, by which, the features of high mutual information with the vector of class labels are selected as follows:

$$maxRelevancy(\boldsymbol{F_s}, \boldsymbol{y}_l) = \frac{1}{card(\boldsymbol{F_s})} \sum_{\boldsymbol{A}_i \in \boldsymbol{F_s}} I(\boldsymbol{A}_i; \boldsymbol{y}_l)$$

where $\boldsymbol{F_s}$ is the selected subset of features, $card(\boldsymbol{F_s})$ is the number of features in $\boldsymbol{F_s}$, and $\boldsymbol{y}_l$ is the vector of class labels.

- A minimum redundancy criterion, by which, the features of low mutual information with other features are selected as follows:

$$minRedundancy(\boldsymbol{F_s}) = \frac{1}{(card(\boldsymbol{F_s}))^2} \sum_{\boldsymbol{A}_i, \boldsymbol{A}_j \in \boldsymbol{F_s}} I(\boldsymbol{A}_i; \boldsymbol{A}_j)$$

$I(\boldsymbol{x};\boldsymbol{y})$ denotes the mutual information between elements $\boldsymbol{x}$ and $\boldsymbol{y}$ defined in terms of their probabilistic density functions as follows:

$$I(\boldsymbol{x};\boldsymbol{y}) = \sum_x \sum_y P(x,y) \log \frac{P(x,y)}{P(x)P(y)}$$

On the other side, there exists another non-classical framework for obtaining a subset of representative and diverse features, i.e. feature clustering. Figure (1.8) shows the general framework of applying clustering for the sake of maximizing diversity among features. Methods under this framework follow the below three-step process:

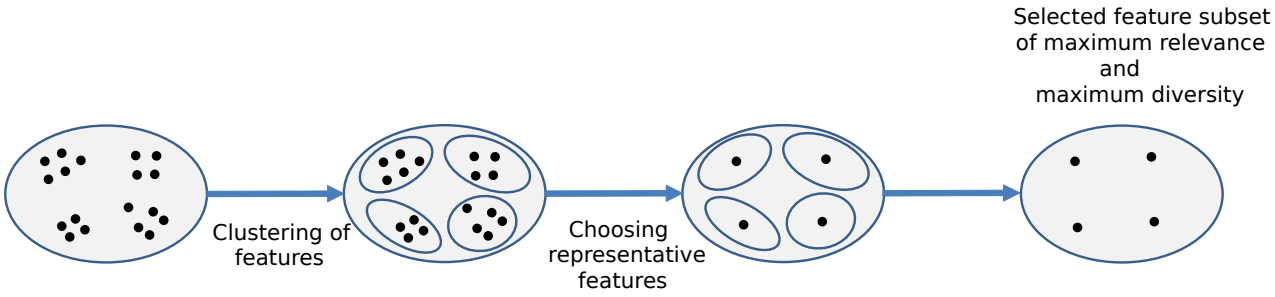


Figure 1.8: A general framework for clustering-based feature selection methods.

- Step 1: A suitable distance measure is chosen for evaluating the similarity or dissimilarity between features.
- Step 2: An appropriate clustering method is applied for grouping similar features.
- Step 3: An appropriate evaluation criterion is applied for choosing the most representative feature out of each cluster.

In [86], Dhillon *et al.* suggested a global criterion for word (feature) clustering and applied it to text classification. It is a novel information theoretic divisive algorithm that minimizes the "within-cluster Jensen-Shannon divergence" while simultaneously

maximizing the "between-cluster Jensen-Shannon divergence". Another supervised method that hierarchically clusters the features (using Ward's method), was proposed by Ienco and Meo [87]. It uses a hybrid method to evaluate the feature subset obtained at each level of the dendrogram[5]. In fact, at each level, a class-distribution preserving measure (Goodman-Kruskal τ [88]) is used to select the most representative features from within each cluster before a classifier evaluates the goodness of the feature subset.
In addition, many clustering-based feature selection methods using information theory were proposed [51, 89, 90]. For instance, one method that uses agglomerative hierarchical clustering builds its own kind of similarity space between features using an information theoretic measure i.e. mutual information [91]. Whereas, Sotoca and Pla [43] build their features dissimilarity space using conditional mutual information. On the other side, Zhao [92] combines the usage of the maximal information coefficient measure with the affinity propagation clustering method to find similar features.

Note that, a feature clustering-based method might select an irrelevant feature as these features are often clustered together. Thus, it is preferable to remove them before feature clustering.

1.7.1 Redundancy Performance Evaluation Metrics

- Representation Entropy (RE) [30, 44]: is an unsupervised metric used to compare the redundancy in obtained feature subsets. It obtains a maximum value when all the eigenvalues become equally important, which means, the level of uncertainty becomes maximum. This indicates that information is evenly distributed among all the principal directions. On the contrary, it obtains a value of zero only when all the information is present along the single principal coordinate direction (all eigenvalues are equal to zero except one).

  The RE of a d-sized feature subset, denoted by H_R, is calculated as follows:

$$H_R = -\sum_{j=1}^{d} \tilde{\lambda}_j \log \tilde{\lambda}_j$$

[5] A multi-level tree diagram called dendrogram. Cutting this dendrogram on a specific level results in one of the possible feature clustering solutions.

where

$$\tilde{\lambda_j} = \frac{\lambda_j}{\sum_{j=1}^{d} \lambda_j}$$

and λ_j is one eigenvalue of the $d \times d$ covariance matrix of the respective feature space.

- Redundancy [59, 82]: another very well-known unsupervised metric that is based on the linear correlation coefficient. It quantifies the strength of pairwise linear relationship between two features. When there is no correlation between two features, then the values of features do not tend to increase or decrease in tandem. It is not ensured that two uncorrelated features are independent, this is since some kind of nonlinear relationship might exist.

  The redundancy measure, denoted by RED, of a feature subset $\boldsymbol{F_s}$ can be calculated as follows:

$$RED(\boldsymbol{F_s}) = \frac{1}{card(\boldsymbol{F_s})(card(\boldsymbol{F_s})-1)} \sum_{\boldsymbol{A}_i, \boldsymbol{A}_j \in \boldsymbol{F_s}, i>j} corr(\boldsymbol{A}_i, \boldsymbol{A}_j)$$

  where $card(\boldsymbol{F_s})$ is the number of features in $\boldsymbol{F_s}$ and $corr(\boldsymbol{A}_i, \boldsymbol{A}_j)$ is linear correlation coefficient between features $\boldsymbol{A}_i$ and $\boldsymbol{A}_j$ calculated as follows:

$$corr(\boldsymbol{A}_i, \boldsymbol{A}_j) = \frac{\sum_n (A_{in} - \mu_{\boldsymbol{A}_i})(A_{jn} - \mu_{\boldsymbol{A}_j})}{\sqrt{\sum_n (A_{in} - \mu_{\boldsymbol{A}_i})^2} \sqrt{\sum_n (A_{jn} - \mu_{\boldsymbol{A}_j})^2}}$$

  where $\mu_{\boldsymbol{A}_i}$ and $\mu_{\boldsymbol{A}_j}$ are the means of the feature vectors $\boldsymbol{A}_i$ and $\boldsymbol{A}_j$ respectively.
  A large value of $RED(\boldsymbol{F_s})$ means that many of the selected features have strong correlation and accordingly it is expected to have redundancy in $\boldsymbol{F_s}$.

1.8 Conclusion

In this chapter, we introduced the general procedure of feature selection as a dimensionality reduction tool. For that, we first had to present the data notations and knowledge representation together with graph data construction and representation.
Afterward, we started by defining the original feature space that can hold relevant, irrelevant, and redundant features, by which, the latter two feature-types are well known to degrade the performance of learning algorithms. As a solution, we highlighted the importance of dimensionality reduction while broadly categorizing it into

feature extraction and feature selection methods. As our focus is on feature selection methods, we only provided a brief overview of some of the widely used feature extraction methods. On the other side, we illustrated the problem of feature selection in the unsupervised, supervised, and semi-supervised learning contexts by detailing a set of representative state of the art score-based ranking methods. In addition, we reviewed the categorization of feature selection into filter, wrapper, and embedded methods according to the evaluation criterion of performance. Finally, we explain the main idea of feature redundancy analysis and state some of its performance evaluation metrics.

As our contribution field is focused on filter-type methods, the next chapter is a detailed review of a powerful family of these methods called Relief-Based Algorithms, by which, we will modify in the later chapters to effectively solve the problem of constrained semi-supervised feature selection.

Chapter 2

Relief-Based Feature Selection

Contents

2.1 Introduction

As elaborated in the previous chapter, the data used in many machine learning and pattern recognition applications is provided with a very large number of features.

Some of which can be irrelevant or redundant. This can lead to over-fitting the learning algorithm, penalizing its performance, and increasing its computational cost. To mitigate such problems, feature selection can be used as a data preprocessing step.

Remembering that according to the evaluation criterion of performance, there are three main categories for feature selection: filter methods [12, 93], wrapper methods, and embedded methods. We are particularly interested in filter Relief-based methods that are simply based on ranking features according to a specific assigned weight. This is motivated by the fact that compared to wrapper methods, Relief avoids any exhaustive combinatorial search and learning algorithm-dependency, hence, it is more efficient. Moreover, it usually performs better than other filter feature selection methods due to the performance feedback of a non-linear classifier when searching for relevant features [94]. At the same time, Relief-based methods are context-aware, taking into account local information to detect dependencies and interactions between features themselves while still providing a global view to select useful features [37].

Therefore, in this chapter, we first explain the original Relief algorithm in section 2.2. Then, by highlighting its strengths and limitations, we show its main variants and extensions in section 2.3. Also, Relief is presented by its four possible interpretations, that are, probabilistic, comprehensible, mathematical and statistical in sections 2.3, 2.4, 2.5 and 2.6 respectively. In addition, Relief is introduced as a margin-based algorithm followed by a set of consequent extensions in section 2.5. Finally, we conclude the chapter in section 2.7.

2.2 The Original Relief Algorithm

Relief was suggested in 1992 by Kira and Rendell [49, 66] as a solution for the problem of selecting a small subset of features that ideally is necessary and sufficient to describe some specific target concept[1] in a supervised context. It is said to be an efficient, simple and practical feature selection solution for two-class real-world classification problems that involve feature interaction without the need of explicitly trying combinations of features.
In addition, Relief that was inspired by instance based-learning [95, 96], calculates a proxy statistic for each feature in order to estimate its "quality" or "relevance" to the target concept. Hence, it is a statistical method that avoids heuristic search. In fact,

[1] A target concept is the endpoint of a specific problem, e.g., it is the class label in classification problems and the predicted value in regression.

in order to calculate these feature statistics, Relief introduced two very important notions, that are, the **nearhit** and the **nearmiss** of a data point. To explain them clearly, we consider the training set $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n, ..., \boldsymbol{x}_N] \in \mathbb{R}^{N \times F}$, where $\boldsymbol{x}_n = \{x_{n1}, ..., x_{ni}, ..., x_{nF}\}$ is the n-th data point in $\boldsymbol{X}$ of N data points characterized by F features with y_n being its corresponding class label.

Definition 2.1. The nearhit of a data point $\boldsymbol{x}_n$, denoted by $\boldsymbol{H}(\boldsymbol{x}_n)$, is defined as its nearest point from the same class and the nearmiss of a data point $\boldsymbol{x}_n$, denoted by $\boldsymbol{M}(\boldsymbol{x}_n)$, is defined as its nearest point from a different class. Figure (2.1a) illustrates these two notions.

Definition 2.2. The feature-statistics, referred to as the features "weights" or "scores" [2], are denoted by a weight vector $\boldsymbol{w} = (w_1, ..., w_i, ..., w_F)^T$ spanning over the F features, where w_i is the relevancy weight corresponding to the i-th feature.

The pseudo-code of Relief is presented below in **Algorithm 2.1**.

Algorithm 2.1 Relief

Input:

- Labeled Training set $\boldsymbol{X}$ of N data points and F features
- Number of iterations T

Output: ***Relevance*** vector

1. Initialize the weight vector to zero $\boldsymbol{w} = (0, 0, ..., 0)$
2. For $t = 1, ..., T$
   (a) Pick randomly a data point $\boldsymbol{x}_n$ from $\boldsymbol{X}$
   (b) Find $\boldsymbol{H}(\boldsymbol{x}_n)$ and $\boldsymbol{M}(\boldsymbol{x}_n)$
   (c) For $i = 1, ..., F$, calculate
   $w_i^{new} = w_i^{old} + \Delta(\boldsymbol{A_i}, \boldsymbol{x}_n, \boldsymbol{M}(\boldsymbol{x}_n))^2 - \Delta(\boldsymbol{A_i}, \boldsymbol{x}_n, \boldsymbol{H}(\boldsymbol{x}_n))^2$
   End For
   End For
3. $\boldsymbol{Relevance} = \boldsymbol{w}/T$
4. The chosen feature set is $\{i | Relevance_i > \tau\}$ where τ is a fixed threshold.

As summarized in **Algorithm 2.1**, Relief iterates over multiple random data points. At each iteration t of T, it selects a data point $\boldsymbol{x}_n$ randomly and without replacement, where T is a user-defined parameter. Then, for the chosen $\boldsymbol{x}_n$, Relief finds the nearhit $\boldsymbol{H}(\boldsymbol{x}_n)$ and nearmiss $\boldsymbol{M}(\boldsymbol{x}_n)$ using squared Euclidean distance. It then updates the value of each feature weight w_i by the value of the difference between the distance from $\boldsymbol{x}_n$ to $\boldsymbol{H}(\boldsymbol{x}_n)$ and from $\boldsymbol{x}_n$ to $\boldsymbol{M}(\boldsymbol{x}_n)$ over this specific feature. For instance, a relevant feature is expected to induce a different value between $\boldsymbol{x}_n$ and $\boldsymbol{M}(\boldsymbol{x}_n)$, thus, when this is true, its corresponding relevancy weight w_i is increased by the value of the difference it induced. Conversely, it is expected from a relevant feature to induce a very similar value (no difference) between $\boldsymbol{x}_n$ and its nearhit $\boldsymbol{H}(\boldsymbol{x}_n)$, therefore, any recorded difference, is subtracted from the corresponding relevancy weight w_i. By meaning, Relief evaluates the relevancy of each feature by calculating its ability to discriminate between near points of two different classes. Hence, a feature that contributes more to this data discrimination is considered more relevant.

The Δ function used in step (c), i.e. the weight-updating step of **Algorithm 2.1**, given by $\Delta(\boldsymbol{A}_i, \boldsymbol{p}_1, \boldsymbol{p}_2)$, calculates the distance between any two data points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ on a specific feature $\boldsymbol{A}_i$.

Definition 2.3. For quantitative features, Δ can be calculated as:

$$\Delta(\boldsymbol{A}_i, \boldsymbol{p}_1, \boldsymbol{p}_2) = \frac{|value(\boldsymbol{A}_i, \boldsymbol{p}_1) - value(\boldsymbol{A}_i, \boldsymbol{p}_2)|}{max(\boldsymbol{A}_i) - min(\boldsymbol{A}_i)} \qquad \in [0,1] \tag{2.1}$$

Similarly, for qualitative features, Δ can be calculated as:

$$\Delta(\boldsymbol{A}_i, \boldsymbol{p}_1, \boldsymbol{p}_2) = \begin{cases} 0, & \text{if} \quad value(\boldsymbol{A}_i, \boldsymbol{p}_1) = value(\boldsymbol{A}_i, \boldsymbol{p}_2) \\ 1, & \text{otherwise} \end{cases} \tag{2.2}$$

Where $value(\boldsymbol{A}_i, \boldsymbol{p})$ is the value of the data point $\boldsymbol{p}$ over the i-th feature. The maximum and minimum values of $\boldsymbol{A}_i$ denoted $max(\boldsymbol{A}_i)$ and $min(\boldsymbol{A}_i)$ are evaluated over the whole set of data points. This normalization ensures that all Δ values are scaled into the interval $[0,1]$ for both qualitative and quantitative features. Note that, the same Δ function is also used to find the nearest neighbors of a considered data point. In fact, the distance between any two data points is simply the sum of their Δ functions over all features.

Figure (2.1) illustrates an example of the weight-updating step with continuous features. In this example, we consider a set of 10 data points divided equally on two

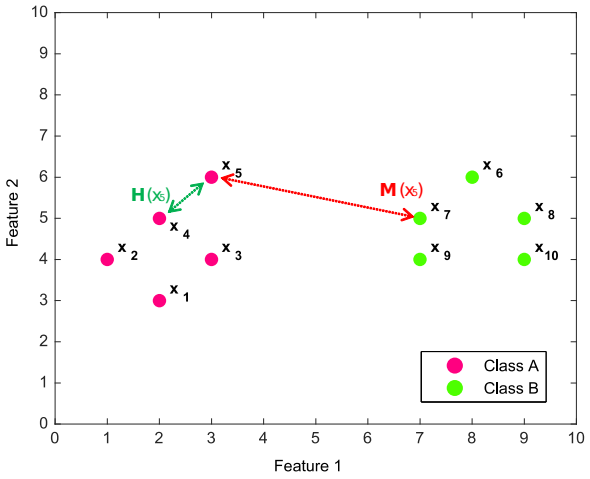
classes (A and B) and defined by two features (Feature 1 and Feature 2) each. The table presented in Figure (2.1b) shows these data points with their feature values, class assignment, and the maximum and minimum values of each feature used for max-min normalization. Figure (2.1a) also shows the 2D plot of these data points with $\boldsymbol{x}_5$ as the considered data point, $\boldsymbol{x}_4$ as its nearhit and $\boldsymbol{x}_7$ its nearmiss. Accordingly, Figure (2.1c) shows how the weights w_1 and w_2, corresponding to Feature 1 and Feature 2 respectively, are updated upon considering the data point $\boldsymbol{x}_5$, its nearhit and its nearmiss; when the value of a feature is different between $\boldsymbol{x}_5$ and its nearhit, this feature loses and its corresponding weight is penalized by the value of the difference $\Big($e.g. $w_1 -= \frac{(3-2)^2}{(max(\text{feature1})-min(\text{feature1}))^2}\Big)$. On the other side, when the value of a feature is different between $\boldsymbol{x}_5$ and its nearmiss, this feature wins and its corresponding weight is increased by the value of the difference $\Big($e.g. $w_1 += \frac{(3-7)^2}{(max(\text{feature1})-min(\text{feature1}))^2}\Big)$.

Finally, after finishing the T iterations, the output (named ***Relevance*** in step (3) of the original Relief algorithm) is a weight vector $\boldsymbol{w}$ divided by T, obtaining an average of the accumulated weight w_i corresponding to each feature $\boldsymbol{A}_i$. This guarantees that all final weights are normalized within the interval $[-1,1]$. Then, each feature having a $Relevance_i$ greater than a given threshold τ is selected. According to the theoretic demonstration by kira and Rendell [49], the value of τ can be statistically evaluated based on Chebyshev's inequality, such that, the probability of choosing an irrelevant feature is as small as possible. However, instead of specifying the threshold value τ, it is often more practical to *a priori* specify the desired size of feature subset (a number) based on the functional, computational, or run time limitations of the learning algorithms to be applied after that. This can be done by ranking the features in the descending order of their weights and then choosing the first 10 features if 10 was the desired feature subset size.

Note that, although the original Relief [66] used Euclidean distance rather than Manhattan distance, later experiments by Kononenko *et al.* [97] showed no significant differences between both distances in the context of Relief, thus, the simplified description using Manhattan distance became a standard [2].

2.2.1 Strengths and Limitations

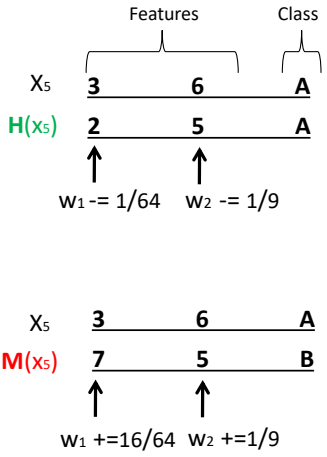
Notably, since Relief avoids heuristic search and does not look for an optimal minimum feature subset size, it is said to be efficient [66]. For instance, compared to the time complexity of an algorithm that does an exhaustive search for detecting interac-



(a) 2D Data Plot

Data point	Feature 1	Feature 2	Class	
X 1	2	3	A	
X 2	1	4	A	
X 3	3	4	A	
X 4	2	5	A	← Nearhit
X 5	3	6	A	← Considered Data point
X 6	8	6	B	
X 7	7	5	B	← Nearmiss
X 8	9	5	B	
X 9	7	4	B	
X 10	9	4	B	
Feature max	9	6		
Feature min	1	3		
Normalization (max-min)	8	3		

(b) A Data table showing the feature values of the considered data point, its Nearhit and its Nearmiss



(c) Weight Updates

Figure 2.1: An illustration of one weight-updating step by Relief over continuous features. (a) Shows the 2D plot of the data points with x_5 as the considered data point, x_4 as its nearhit and x_7 its nearmiss. (b) Shows the data points table with the maximum and minimum value of each feature used for max-min normalization. (c) Shows how the weights w_1 and w_2 corresponding to Feature 1 and Feature 2 respectively are updated.

tions between all feature pairs i.e. $O(2^F)$, Relief has a time complexity of $O(TNF)$, or $O(NF)$ whenever $T < N$. Thus, it requires linear time in terms of the number of features and training data points. In addition, Relief is non-parametric, which means, it ranks the features without making any assumptions regarding the population distribution or sample size [2]. Moreover, Relief can be seen as an any-time ranking algorithm. This means that it can be stopped at any iteration and obtain a relevancy-ranking of features, yet, with more time or data its results can be improved. Relief has been also categorized as non-myopic, which means it can evaluate the relevance of features in the context of other features (detects feature interactions) [98].
However, the original Relief is basically limited to two-class classification problems. Although, Kira and Rendell [66] suggested that, as future work, the algorithm can be easily extended to solve multi-class classification problems by splitting them into a series of two-class problems, this solution was considered unsatisfactory and inefficient [99]. Relief also has no mechanism to work with missing data or feature redundancy. In fact, the original Relief analysis suggests that its performance might be penalized by insufficient training iterations (i.e. not a large enough T). Most importantly, in 1994, Kononenko identified that Relief was susceptible to noise interfering with the selection of nearest neighbors [4].

2.3 Basic Variants and Extensions of Relief with Probabilistic Interpretation

Now that we detailed the aspects and notions of the basic original Relief algorithm, it is important to note that it gained a big deal of interest between filter-type feature selection methods leading to a series of extensions, interpretations, and modifications. In this section, we go through a concise review of the Relief-Based Algorithms, known recently as RBAs [2].

Initially, in the context of instance-based learning, kononenko [4] derived a **probabilistic interpretation** of the feature weights obtained by Relief.

Definition 2.4. The probabilistic interpretation considers that a good feature A_i should:

- induce a high probability of having a different value over a data point $\boldsymbol{x}_n$ and its own nearmiss $\boldsymbol{M}(\boldsymbol{x}_n)$.
- induce a low probability of having a different value over a data point $\boldsymbol{x}_n$ and its

own nearhit $\boldsymbol{H}(\boldsymbol{x}_n)$.

Thus, the weight w_i of the feature $\boldsymbol{A}_i$ is obtained as an approximation of the difference between the above probabilities:

$$\begin{aligned} w_i = & P\Big(value(\boldsymbol{A}_i, \boldsymbol{x}_n) \neq value(\boldsymbol{A}_i, \boldsymbol{M}(\boldsymbol{x}_n))\Big) \\ & - P\Big(value(\boldsymbol{A}_i, \boldsymbol{x}_n) \neq value(\boldsymbol{A}_i, \boldsymbol{H}(\boldsymbol{x}_n))\Big) \end{aligned} \tag{2.3}$$

However, from this probabilistic point of view, it is considered that a larger number of iterations T, i.e the number of data points used for approximating probabilities, provides a more reliable approximation. At the same time, T should not exceed the number of data points available in $\boldsymbol{X}$, i.e $T \leq N$.

2.3.1 Robustness

By following the drawbacks of Relief, Kononenko stated that the selection of nearest neighbors with respect to important features is one of the most important steps in Relief [4]. However, when selecting them in a context of irrelevant (redundant and noisy) features, the process can become unreliable and negatively affect the probability estimations. Therefore, kononenko proposed the first modification of Relief, called ReliefA, as a solution. It is an extended more robust version of Relief that aims at increasing the reliability of probability approximations. For that, ReliefA searches for K-nearhits and K-nearmisses instead of only one nearhit and one nearmiss. In fact, ReliefA can be seen as a functionality-based extension, i.e, averaging the contributions of K-nearhits/misses.

Definition 2.5. The K-nearhits of $\boldsymbol{x}_n$ from the same class are denoted by $K\boldsymbol{H}(\boldsymbol{x}_n)$: $\{\boldsymbol{H}_1(\boldsymbol{x}_n), \boldsymbol{H}_2(\boldsymbol{x}_n)$, ..., $\boldsymbol{H}_K(\boldsymbol{x}_n)\}$ and the K-nearmisses of $\boldsymbol{x}_n$ from the opposite class in a two-class problem are denoted by $K\boldsymbol{M}(\boldsymbol{x}_n) : \{\boldsymbol{M}_1(\boldsymbol{x}_n), \boldsymbol{M}_2(\boldsymbol{x}_n)$, ..., $\boldsymbol{M}_K(\boldsymbol{x}_n)\}$. Where K is a user defined parameter representing the number of closest points to $\boldsymbol{x}_n$ to be considered.

However, the number of considered nearest neighbors K should be kept relatively small. This is due to the fact that as K increases from 1 approaching N, the condition of using the "near" data points for feature weight estimation is eliminated. This was derived by Robnik-Šikonja and Kononenko [37] who showed that without the "near" requirement, Relief weights might lose their ability to consider local information and hence the ability to detect feature interaction. In fact, the power of RBAs lies in their

ability to efficiently exploit information locally while still being able to provide a global view with context awareness.

It was noted that using ReliefA instead of Relief improves the estimates of features for both free of noise and noisy datasets, however, a more significant improvement is obtained when the data is noisy [4].

2.3.2 Incomplete Data

Following the usage of K-nearest neighbors, five extensions were proposed to ReliefA to deal with missing values and multi-class problems, known as ReliefB through ReliefF [4]. Three of these extensions that are ReliefB, ReliefC, and ReliefD, were proposed in order to deal with incomplete data or missing values.
In fact, ReliefB handles missing values by modifying the output of the corresponding Δ function as follows:

- if at least one of the two data points has an unknown value on a specific feature $\boldsymbol{A}_i$:

$$\Delta(\boldsymbol{A}_i, \boldsymbol{p}_1, \boldsymbol{p}_2) = 1 - \frac{1}{\#ofvalues_{\boldsymbol{A}_i}}$$

  where $\#ofvalues_{\boldsymbol{A}_i}$ is the number of possible values for feature $\boldsymbol{A}_i$.

On the other hand, ReliefC differs from ReliefB by disregarding the contributions of the differences that are calculated from data points of at least one unknown value on a specific feature through suitable normalization. This is done with the assumption that when enough training data is provided, the resulting feature weights should converge to their correct estimate values.

However, experiments showed that ReliefD performed the best in the context of incomplete and noisy data. For that, when ReliefD encounters an unknown value of a data point on a specific feature, it replaces the output of the corresponding Δ function by the class-conditional probability that these two data points have different values over this specific feature as follows:

- if one data point of the two has an unknown value (e.g. value of $\boldsymbol{p}_1$ over $\boldsymbol{A}_i$ is unknown):

$$\Delta(\boldsymbol{A}_i, \boldsymbol{p}_1, \boldsymbol{p}_2) = 1 - P(value(\boldsymbol{A}_i, \boldsymbol{p}_2)|class(\boldsymbol{p}_1))$$

- if both of the data points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ have unknown values over $\boldsymbol{A}_i$:

$$\Delta(\boldsymbol{A}_i, \boldsymbol{p}_1, \boldsymbol{p}_2) = 1 - \sum_{value_{\boldsymbol{A}_i}}^{\#ofvalues_{\boldsymbol{A}_i}} P(value_{\boldsymbol{A}_i}|class(\boldsymbol{p}_1)) \times P(value_{\boldsymbol{A}_i}|class(\boldsymbol{p}_2))$$

Where $\#ofvalues_{\boldsymbol{A}_i}$ is the number of possible values for feature $\boldsymbol{A}_i$ and the values of conditional probabilities are approximated using the relative frequencies over the training data.

On the contrary, Relieved-F suggested by kohavi *et al.* [100] handles missing data by simply setting the difference between two unknown values to 0 and the difference between an unknown and any other known value to one.

2.3.3 Multi-Class Problems

Kira and Rendell [49, 66] proposed that the original Relief algorithm can be applied in the context of three or more classes through splitting the domain or problem into a series of two-class ones. However, this was proved to be inefficient and unsatisfactory [97]. Therefore, as more practical solutions, the two other extensions, namely ReliefE and ReliefF, were proposed following ReliefD to efficiently deal with the problem of having more than two classes [4]. As for ReliefE, the straightforward generalization of Relief, it simply modifies the definition of nearmiss to be the one nearest point from any "other" class with respect to a data point $\boldsymbol{x}_n$.

On the other side, ReliefF that includes both of the suggested modifications applied in ReliefA and ReliefD, is the most common and well-known variant of the original Relief algorithm nowadays. For instance, in order to deal with multi-class problems, ReliefF finds a nearmiss from each different class instead of finding only one nearmiss from the opposite class in the case of two-class problems or from any different class as in ReliefE in case of multi-class problems. Then, the contributions of each nearmiss from each "other" class are averaged based on the prior probability of each class and used to update weight estimates. By meaning, the algorithm evaluates the ability of features to separate all pairs of classes regardless of their number and regardless of which two classes are the closest to each other. Moreover, as the capabilities of ReliefF include those of ReliefA, it can also choose K-nearmisses from each 'different' class and K-nearhits from the same class of a considered data point.

Definition 2.6. The K-nearhits of $\boldsymbol{x}_n$ from the same class are defined as in the previous definition **Definition 2.5** without being changed. However, the K-nearmisses of $\boldsymbol{x}_n$ from each "other" class are now denoted by $K\boldsymbol{M}(\boldsymbol{x}_n, c) : \{\boldsymbol{M}_1(\boldsymbol{x}_n, c), \boldsymbol{M}_2(\boldsymbol{x}_n, c), ... \boldsymbol{M}_K(\boldsymbol{x}_n, c)\}$. Where K is a user defined parameter representing the number of closest points to $\boldsymbol{x}_n$ to be considered and c denotes the class to which each nearmiss belongs.

Finally, kononenko [4] concluded that as the number of iterations or considered data points, i.e. T, approaches the total number of data points N, the quality of the weight estimates becomes more reliable. Hence, it became the default to set $T = N$, where each data point is considered once without replacement. In this regard, the Relieved-F [100], mentioned before, can be seen as a very straightforward extension to ReliefF. It is a deterministic algorithm that uses all data points and all possible nearhits and possible nearmisses of each point. For instance, if there is a tie between two points on being equally "near" to a considered data point, their contributions are averaged instead of choosing one of them randomly. Kohavi *et al.* [100] claimed that although this way requires only as much time as the standard Relief algorithm with $T = N$, it would provide the result expected when it is allowed to run for an infinite amount of time.

Definition 2.7. The weight updating step in ReliefF can be given by:

$$\begin{aligned} w_i^{new} = w_i^{old} - \tfrac{1}{(T \cdot K)} \textstyle\sum_{j=1}^{K} \Delta(A_i, \boldsymbol{x}, \boldsymbol{H}_j) + \\ \tfrac{1}{(T \cdot K)} \textstyle\sum_{c \neq class(\boldsymbol{x})} \left[\tfrac{P(c)}{1 - P(class(\boldsymbol{x}))} \textstyle\sum_{j=1}^{K} \Delta\big(A_i, \boldsymbol{x}, \boldsymbol{M}_j(c)\big) \right] \end{aligned} \tag{2.4}$$

Where $P(c)$ represents the prior probability of class c (estimated from the training set) and $1 - P(class(\boldsymbol{x}))$ represents the sum of probabilities for the misses' classes. Note that, the normalization using these prior class probabilities is a necessity. This is due to the fact that using K-nearmisses from each "other" class would tend to unreliably reflect an exaggerated influence of classes with a small number of elements. Accordingly, the pseudo-code of ReliefF is presented below in **Algorithm 2.2**.

Note that, later in 1996, Kononenko *et al.* [99] suggested an adaptation of ReliefF to regression problems called RReliefF. In such cases where the target concept (i.e a discrete variable in classification problems) is a continuous variable (real-valued function), the notions of nearhit and nearmiss cannot be explicitly used. This is due to the impossibility of realizing the exact knowledge of whether two data points belong to the same class or not. Thus, the probability that the predicted value of two data points is different was introduced instead [101].

Algorithm 2.2 ReliefF

Input:

- Labeled Training set $\boldsymbol{X}$ of N data points and F features
- Number of iterations T

Output: Weight vector $\boldsymbol{w}$

1. Initialize the weight vector to zero $\boldsymbol{w} = (0,0,...,0)$
2. For $t = 1,...,T$
   (a) Pick randomly a data point $\boldsymbol{x}$ from $\boldsymbol{X}$
   (b) Find K-nearhits $K\boldsymbol{H}(\boldsymbol{x})$
   (c) For each class $c \neq class(\boldsymbol{x})$
   find K-nearmisses $K\boldsymbol{M}(\boldsymbol{x})$
   End For
   (d) For $i = 1,...,F$, calculate
   $w_i^{new} = w_i^{old} - \frac{1}{(T \cdot K)} \sum_{j=1}^{K} \Delta(A_i, \boldsymbol{x}, \boldsymbol{H}_j) +$
   $\frac{1}{(T \cdot K)} \sum_{c \neq class(\boldsymbol{x})} \left[\frac{P(c)}{1-P(class(\boldsymbol{x}))} \sum_{j=1}^{K} \Delta\big(A_i, \boldsymbol{x}, \boldsymbol{M}_j(c)\big) \right]$
   End For
   End For
3. The chosen feature set is $\{i | w_i > \tau\}$ where τ is a fixed threshold

2.4 Concept of Change Interpretation

So far, the probabilistic interpretation of Relief weights was well suited to hold and explain a set of statistical and class-discriminative properties and extensions of RBAs. However, it fails when trying to find a human-comprehensible explanation of Equation (2.3). For instance, subtraction of probabilities and the negated feature similarity (different values on a feature) are difficult to understand by human experts in real-world applications. Therefore, another interpretation (based on a theoretical analysis) of the Relief's weights or estimates was introduced by Robnik-Šikonja and Kononenko [5]. It is said to be more understandable and comprehensible than the probabilistic one.

In fact, Robnik-Šikonja and Kononenko suggested that the feature weights, assigned by Relief, can be interpreted as their ability to explain a change in the value of the endpoint (class or predicted value). However, this interpretation is proposed in the context of a problem space that is densely covered with data points to analyze the behavior of Relief. By meaning, it assumes that when the number of data points approaches infinity, Relief's weights (quality estimates of each feature) converge and can be interpreted as the ratio between the number of explained changes in the class labels[2] and the number of considered data points. If it is possible to explain a certain change in the endpoint value in multiple different ways, all of these ways share credit for it in the feature weight. Also, if two or more features are involved in one of the explanation ways, all of these features gain the same credit to their relevancy weight.

An example which illustrates the "concept of change interpretation" was detailed in [37] and [5]. We use the same example to better explain the idea. Thus, Table 2.1 shows a boolean domain where the predicted value (class) of a data point is determined by the following expression: Class=$(\boldsymbol{A}_1 \cdot \boldsymbol{A}_2) + (\boldsymbol{A}_1 \cdot \boldsymbol{A}_3)$.

Table 2.1: A table showing the data points and the responsibility of each feature in changing the outcome of the boolean problem defined by Class = $(A1 \cdot A2) + (A1 \cdot A3)$. Example proposed by Robnik-Šikonja and Kononenko [5, 37].

Data point	Features			Class	Features holding Responsibilty of Class Change	Score		
	$\boldsymbol{A}_1$	$\boldsymbol{A}_2$	$\boldsymbol{A}_3$			$\boldsymbol{A}_1$	$\boldsymbol{A}_2$	$\boldsymbol{A}_3$
x_1	1	1	1	1	$\boldsymbol{A}_1$	1/8	0	0
x_2	1	1	0	1	$\boldsymbol{A}_1$ or $\boldsymbol{A}_2$	0.5/8	0.5/8	0
x_3	1	0	1	1	$\boldsymbol{A}_1$ or $\boldsymbol{A}_3$	0.5/8	0	0.5/8
x_4	1	0	0	0	$\boldsymbol{A}_2$ or $\boldsymbol{A}_3$	0	0.5/8	0.5/8
x_5	0	1	1	0	$\boldsymbol{A}_1$	1/8	0	0
x_6	0	1	0	0	$\boldsymbol{A}_1$	1/8	0	0
x_7	0	0	1	0	$\boldsymbol{A}_1$	1/8	0	0
x_8	0	0	0	0	$\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ or $\boldsymbol{A}_1$ and $\boldsymbol{A}_3$	1/8	0.5/8	0.5/8
					Total Score	0.75	0.1875	0.1875

By analyzing the first row of Table 2.1, we can say that $\boldsymbol{A}_1$ takes the responsibility of changing the value of the class label. This is clear since changing its value alone from 1 to 0 affects the value of the class similarly. Whereas, changing the value of $\boldsymbol{A}_2$ or $\boldsymbol{A}_3$ alone leaves the class value unaltered. In addition, by analyzing the second row, it can be noticed that changing the values of either $\boldsymbol{A}_1$ or $\boldsymbol{A}_2$ can be responsible for altering the class label. Thus, in this case, changing $\boldsymbol{A}_1$ and changing $\boldsymbol{A}_2$ represent

[2] An explained change of concept is a change in the class label that can be justified by a feature value change.

two possible ways of "change in concept" and they share the responsibility for that. It is also possible to similarly analyze rows three to seven to decide which feature is responsible for the class change. However, the analysis of the eighth row shows that changing only one feature is not enough for a class change. Nevertheless, changing A_1 and A_2 or A_1 and A_3 changes the class label. Hence, a minimum of two changes is required to change the class and the responsibility score goes to both of the feature pairs A_1 and A_2 or A_1 and A_3. Since there are 8 data points in this example and they all have an equal probability of occurrence, each responsibility score is divided by 8. For instance, when feature A_1 alone was responsible for the class change (rows 1, 5, 6 and 7) it gains a value of $1/8$ for each. On the other side, when it shares the responsibility with another feature each of them gets a score of $0.5/8$. Finally, as can be seen in the Total Score row of Table 2.1, feature A_1 gets a total score of 0.75 and both A_2 and A_3 get a score of 0.1875 each.

To validate this conceptual example, Robnik-Šikonja and Kononenko [5] added five random binary features next to the three relevant ones that were discussed above. Then, they empirically showed that the feature weights obtained by Relief and ReliefF for this problem converge to 0.75 for A_1 and 0.187 for A_2 and A_3 when the number of data points is large. Also, a detailed analysis, derivation, and proof of this interpretation is presented in [5] with a complete description of how it differs between Relief and ReliefF.

2.5 Margin Notion in Relief-Based Feature Selection

In 2004, Gilad-Bachrach *et al.* [102] presented the Relief-family weight updating function as a "Maximal Margin" or "Large Margin" objective function to be optimized.

2.5.1 Margin General Definition and Types

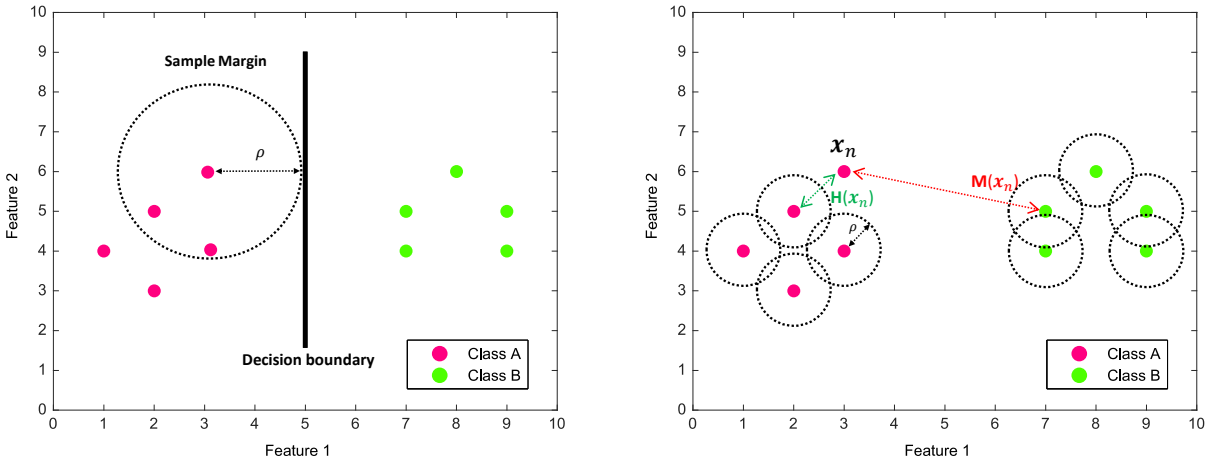
Intuitively speaking, the margin is a geometric measure that evaluates the confidence of a classifier when making its decision. Moreover, it plays a crucial role in the machine learning and pattern recognition domains and similarly to other tools and techniques in such domains, the margin was initially defined in the supervised context [103–105]. It was also noted that it can be used for theoretic generalization bounds and as guidelines for algorithmic design [104]. Two approaches were declared for defining the margin of a data point with respect to a classification rule [106]:

- The **sample-margin**, i.e. the more common approach, measures the distance be-

tween a data point and the decision boundary induced by a specific classifier. For example, Support Vector Machines [76] include a classification algorithm that represents data examples as points in space, mapped so that the data points belonging to different classes are separated by a clear gap (sample-margin) that is as wide as possible. Figure (2.2a) shows how the sample margin quantifies the distance a data point can travel before it hits the decision boundary.

- The **hypothesis-margin**, i.e. the alternative definition, requires a distance measure between data points. It defines the margin as the difference between the distance from a data point to its nearest point of alternative class (defined by Relief as **nearmiss**) and the distance to its nearest point of the same class (defined by Relief as **nearhit**).
  Figure (2.2b) provides an illustration of the hypothesis-margin. By meaning, it is the largest distance a data point can travel within the feature space without altering the dataset's labeling structure, and thus without affecting the label prediction of a new arriving point. This is done, such that, drawing a circle of a radius equal to the margin (denoted ρ) around each data point and allowing each of these points to freely change its position within its corresponding circle, does not alter the assigned label to the new data point. This shows that the hypothesis-margin measures the stability to trivial changes in the data points locations.

Although using the sample-margin looks initially more natural, it was shown in [106] that it can be unstable when using the first nearest neighbor (1-NN) margin. This is due to being very sensitive to the smallest relocations of the data points (a small position-change can lead to a dramatic alteration in the decision boundary of the sample-margin). Moreover, it was proved that the hypothesis-margin lower bounds the sample-margin and has a lower computational cost for high-dimensional data sets [106, 107]. Which means, in the case of 1-NN, a large hypothesis-margin ensures a large sample-margin in addition to being simple and easy to compute.
Therefore, noticing the resemblance between the nearest neighbor notions of the weight-updating rule (proposed in the original Relief) and the hypothesis-margin definition, made it natural to utilize margin maximization in feature selection. This is applied under the assumption that a feature's ability to discriminate between data points with respect to a specific classification rule can be measured by its contribution to the maximization of this hypothesis-margin [102].



(a) Sample-Margin (b) Hypothesis-Margin

Figure 2.2: The two types of nearest neighbor margin denoted by ρ. We show how to measure the margin (radius of dotted circles) with respect to a new data point considering a set of labeled data points; (a) The sample-margin i.e. the distance between the considered data point and the decision boundary. (b) The hypothesis-margin i.e. the maximum distance the data points can travel without altering the label of the new data point.

2.5.2 Mathematical Interpretation

Inspired by the work done in [102] where the margin notion was introduced to Relief family and the maximization of the averaged margin was empirically observed, Sun and Li [6] proposed a mathematical interpretation of Relief from its optimization point of view. Before we get to show Relief from its optimization perspective, we generally define the margin and the weighted-margin independently of any specific distance measure.

Definition 2.8. The hypothesis-margin of a data point $\boldsymbol{x}_n$ denoted by $\rho(\boldsymbol{x}_n)$ with respect to the training data $\boldsymbol{X}$ is the difference between the distance from $\boldsymbol{x}_n$ to its nearmiss and the distance from $\boldsymbol{x}_n$ to its nearhit[3] given by:

$$\rho(\boldsymbol{x}_n) = \Delta(\boldsymbol{x}_n, \boldsymbol{M}(x_n)) - \Delta(\boldsymbol{x}_n, \boldsymbol{H}(x_n)) \tag{2.5}$$

where $\Delta(\boldsymbol{p}_1, \boldsymbol{p}_2)$ is defined as the generalized distance function between any two data points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ calculated in the space of all features F with $p \in \mathbb{R}$ and $\Delta(\boldsymbol{A}_i, \boldsymbol{p}_1, \boldsymbol{p}_2)$

[3]Recalling that the nearhit and nearmiss notions are declared in **Definition 2.1** and illustrated in Figure (2.1a).

from **Definition 2.3** as follows:

$$\Delta(\boldsymbol{p}_1, \boldsymbol{p}_2) = \left(\sum_{i=1}^{F} |\Delta(\boldsymbol{A}_i, \boldsymbol{p}_1, \boldsymbol{p}_2)|^p \right)^{\frac{1}{p}} \tag{2.6}$$

Note that, $\rho(\boldsymbol{x}_n)$ is only positive when $\boldsymbol{x}_n$ is correctly classified by 1-NN classifier, which means, when the distance from $\boldsymbol{x}_n$ to its nearmiss is larger than the distance from $\boldsymbol{x}_n$ to its nearhit.

Thus, the overall hypothesis-margin over the training set $\boldsymbol{X}$ can be computed as:

$$\rho(\boldsymbol{X}) = \sum_{n=1}^{N} \rho(\boldsymbol{x}_n) \tag{2.7}$$

As the ability of a feature to discriminate between data points can be evaluated by how much it contributes to the margin's maximization, this contribution can be represented and quantified by the same weight vector $\boldsymbol{w} = (w_1, ..., w_i, ..., w_F)^T$ spanning over the F features defined in **Definition 2.2** [102]. Thus, one natural idea is to scale each feature by a non-negative vector $\boldsymbol{w}$ to obtain a weighted feature space such that a margin-based function in this induced feature space is maximized [6, 102]. Note that, applying margin-based feature selection in a weighted space is motivated by the fact that the nearest neighbors in the original space are highly unlikely to be the same in the weighted feature space [6, 108].

Definition 2.9. Considering a weight vector $\boldsymbol{w} = (w_1, ..., w_i, ..., w_F)^T$, the weighted hypothesis-margin of a data point $\boldsymbol{x}_n$ with respect to the training set $\boldsymbol{X}$ is given by:

$$\rho(\boldsymbol{x}_n, \boldsymbol{w}) = \Delta_{\boldsymbol{w}}(\boldsymbol{x}_n, \boldsymbol{M}(\boldsymbol{x}_n)) - \Delta_{\boldsymbol{w}}(\boldsymbol{x}_n, \boldsymbol{H}(\boldsymbol{x}_n)) \tag{2.8}$$

where $\Delta_{\boldsymbol{w}}(\boldsymbol{p}_1, \boldsymbol{p}_2)$ is defined as the generalized distance function between any two data points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ calculated in the weighted space of all features in F with $p \in \mathbb{R}$ as follows:

$$\Delta_{\boldsymbol{w}}(\boldsymbol{p}_1, \boldsymbol{p}_2) = \left(\sum_{i=1}^{F} |w_i \Delta(\boldsymbol{A}_i, \boldsymbol{p}_1, \boldsymbol{p}_2)|^p \right)^{\frac{1}{p}} \tag{2.9}$$

According to Equations (2.8) and (2.9), each time the margin is calculated with respect to a data point $\boldsymbol{x}_n$ of F features, the weight vector $\boldsymbol{w}$ contributes to this calculation. Taking (2.8) into account, it can be generalized to calculate the weighted hypothesis-margin over all the given data points. Consequently, the final weight vector of dimension F shows the impact of each feature in enlarging the margin, which is

considered the relevancy score of the feature in terms of original Relief notions.

Thus, the weighted hypothesis-margin over the training set $\boldsymbol{X}$ is computed as:

$$\rho(\boldsymbol{X}, \boldsymbol{w}) = \sum_{n=1}^{N} \rho(\boldsymbol{x}_n, \boldsymbol{w}) \tag{2.10}$$

It is clear that Equation (2.10) is the summation of the weighted hypothesis margins over the whole training set. So, the key idea is that as a feature contributes more in enlarging the overall weighted margin, it gets higher weights, and will consequently have a higher chance of being selected.

Changing the value of p in Equations (2.6) and (2.9) defines different margin-based feature selection algorithms. If it is set to 1 i.e l_1-norm, deriving Manhattan distance, the standardized Relief algorithm is obtained [6, 50, 94]; if p is set to 2 i.e l_2-norm deriving Euclidean distance, the resulting algorithm is Simba [102]; also, if p is set to 2 but the squared Euclidean distance is used, the resulting algorithm is the original Relief initially defined in [66].

As a side note, in the context of face recognition, where an enhanced block-based face descriptor is used to represent face images, Moujahid and Dornaika [109] proposed a modified Relief algorithm (m-RELIEF) that uses a weighted χ^2 distance to select features (i.e. in this case histograms) instead of using the classical Euclidean distance.

2.5.2.1 Optimization Approach to Relief Algorithm

Sun and Li pointed out that the reason behind widely depending on heuristic-search approaches for feature selection is the difficulty of defining an objective function, by which, well-established optimization techniques can optimize [6, 50, 94]. However, they noticed that although Relief is a feature-weighting algorithm that allows working with soft feature-weights[4], it did not have a clearly defined objective function to be optimized. Therefore, they defined the standardized Relief from its optimization point of view over a margin-based objective function using Manhattan distance (i.e $p = 1$ in Equations (2.6) and (2.9)). This new interpretation enabled viewing Relief as an online feature-weighting algorithm. The latter makes use of the performance of a

[4]Soft weights means that weights assigned to each feature are real-valued instead of being binary indicators of whether a feature is selected or not. The problem of defining an optimizable objective function in feature selection can be somehow mitigated by using feature-weighting strategies [8].

highly non-linear nearest neighbor classifier to learn discriminant information locally with the aim of identifying useful features. For that, it solves a simple convex problem in a closed-form[5] which justifies its simplicity and effectiveness.

Accordingly, the hypothesis-margin of a data point $\boldsymbol{x}_n$ denoted by $\rho(\boldsymbol{x}_n)$ is defined as:

$$\rho(\boldsymbol{x}_n) = |\boldsymbol{x}_n - \boldsymbol{M}(\boldsymbol{x}_n)| - |\boldsymbol{x}_n - \boldsymbol{H}(\boldsymbol{x}_n)| \tag{2.11}$$

Where the used distance function $|.|$ is the l_1-norm (or Manhattan distance) defined as:

$$|\boldsymbol{v}| = \sum_{i=1}^{F} |v_i| = |v_1| + ... + |v_i| ... + |v_F| \tag{2.12}$$

The weighted hypothesis-margin of $\boldsymbol{x}_n$ with respect to $\boldsymbol{X}$ is:

$$\rho(\boldsymbol{x}_n, \boldsymbol{w}) = |\boldsymbol{x}_n - \boldsymbol{M}(\boldsymbol{x}_n)|_{\boldsymbol{w}} - |\boldsymbol{x}_n - \boldsymbol{H}(\boldsymbol{x}_n)|_{\boldsymbol{w}} \tag{2.13}$$

where,

$$|\boldsymbol{v}|_{\boldsymbol{w}} = |\boldsymbol{w} \circ \boldsymbol{v}| = \sum_{i=1}^{F} w_i |v_i| = w_1|v_1| + ... + w_i|v_i| + ... + w_F|v_F| \tag{2.14}$$

Knowing that $\boldsymbol{w} \geq \boldsymbol{0}$ and $\circ$ is the Hadamard product.

Definition 2.10. The margin-based objective function from the optimization point of view is obtained by scaling each feature such that the averaged hypothesis-margin in the weighted feature space can be maximized as follows:

$$\begin{aligned} &\max_{\boldsymbol{w}} \textstyle\sum_{n=1}^{N} \rho(\boldsymbol{x}_n, \boldsymbol{w}) \\ &= \max_{\boldsymbol{w}} \textstyle\sum_{n=1}^{N} \left(\sum_{i=1}^{F} w_i |\boldsymbol{x}_{ni} - \boldsymbol{M}(x_n)_i| - \sum_{i=1}^{F} w_i |\boldsymbol{x}_{ni} - \boldsymbol{H}(x_n)_i| \right) \\ &\text{s.t. } \|\boldsymbol{w}\|_2^2 = 1 \text{ and } \boldsymbol{w} \geq \boldsymbol{0} \end{aligned} \tag{2.15}$$

where the weight vector $\boldsymbol{w} \geq \boldsymbol{0}$ ensures that it induces a distance measure. In fact, it is intuitive that the chosen weight vector should induce a positive value when features are relevant. Also, $||\boldsymbol{w}||_2^2 = 1$ ensures that the vector is not maximized without bounds.

Definition 2.11. Let $\boldsymbol{z} = (z_1, ..., z_i, ..., z_F)^T$ be a vector that holds the hypothesis-

[5] An equation is said to be a closed-form solution if it solves a given problem in terms of functions and mathematical operations from a given generally-accepted set. For example, an infinite sum would generally not be considered a closed-form [110].

margin induced by each feature when considering all data points in $\boldsymbol{X}$ as follows:

$$z_i = \sum_{n=1}^{N} |x_{ni} - M(\boldsymbol{x}_n)_i| - |x_{ni} - \boldsymbol{H}(x_n)_i| \tag{2.16}$$

For clarity, we express $\boldsymbol{z}$ in the following notation:

$$\boldsymbol{z} = \sum_{n=1}^{N} \left(\begin{pmatrix} |x_{n1} - M(x_n)_1| \\ ... \\ |x_{nF} - M(x_n)_F| \end{pmatrix} - \begin{pmatrix} |x_{n1} - H(x_n)_1| \\ ... \\ |x_{nF} - H(x_n)_F| \end{pmatrix} \right) = \begin{pmatrix} z_1 \\ ... \\ z_F \end{pmatrix} \tag{2.17}$$

Definition 2.12. Considering $\boldsymbol{z}$, the simplified margin-based objective function to be maximized is given by:

$$\max_{\boldsymbol{w}} \quad \boldsymbol{w}^T \boldsymbol{z}, \quad \text{s.t.} \quad \|\boldsymbol{w}\|_2^2 = 1, \quad \boldsymbol{w} \geq \boldsymbol{0} \tag{2.18}$$

By using the Lagrangian function on (2.18), the equation can be expressed as:

$$\mathcal{L}(\boldsymbol{w}, \lambda, \boldsymbol{\mu}) = -\boldsymbol{w}^T \boldsymbol{z} + \lambda(\boldsymbol{w}^T \boldsymbol{w} - 1) - \boldsymbol{\mu}^T \boldsymbol{w} \tag{2.19}$$

where $\lambda > 0$ and $\boldsymbol{\mu} \geq \boldsymbol{0}$ are the Lagrangian multipliers.
Now by taking the derivative of $\mathcal{L}$ with respect to $\boldsymbol{w}$ we obtain:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = -\boldsymbol{z} + 2\lambda \boldsymbol{w} - \boldsymbol{\mu} \tag{2.20}$$

And satisfying stationarity condition by solving $\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = 0$ we get:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = 0 \implies \boldsymbol{w} = \frac{1}{2\lambda}(\boldsymbol{z} + \boldsymbol{\mu}) \tag{2.21}$$

To derive a closed-form solution for $\boldsymbol{w}$, it is assumed that there exists an $i : 1 \leq i \leq F$ where $z_i > 0$, when this assumption is not true, i.e, when $\boldsymbol{z} \leq 0$, the distance between a data point and its nearhit is simply greater than the distance between it and its nearmiss, which negatively affects the performance of neighborhood-based learning algorithms. However, using this assumption it can be proved by contradiction that $\lambda > 0$. Consider the case when $\lambda < 0$ while holding the assumption that $z_i > 0$, then $z_i + \mu_i > 0$. It is then obtained that $w_i < 0$ which contradicts with the initial constraint $\boldsymbol{w} \geq \boldsymbol{0}$.

According to Sun and Li [6], the Karush Kuhn Tucker (KKT [111]) complementarity

condition: $\boldsymbol{\mu}^T \boldsymbol{w} = 0$ is used to verify the three following cases:

1. $z_i = 0 \Rightarrow \mu_i = 0 \Rightarrow w_i = 0$.
2. $z_i > 0 \Rightarrow z_i + \mu_i > 0 \Rightarrow w_i > 0 \Rightarrow \mu_i = 0$.
3. $z_i < 0 \Rightarrow \mu_i > 0 \Rightarrow w_i = 0 \Rightarrow z_i = -\mu_i$.

Finally, the optimal solution can be calculated in a closed-form as follows:

$$\begin{aligned} &\boldsymbol{w}^* = \boldsymbol{w} / \|\boldsymbol{w}\|_2 = (\boldsymbol{z})^+ / \|(\boldsymbol{z})^+\|_2 \\ &\text{where, } \boldsymbol{z} \text{ is given by Equation (2.17)} \\ &\text{and } (\boldsymbol{z})^+ = [\max(z_1, 0), ..., \max(z_F, 0)]^T \end{aligned} \tag{2.22}$$

By comparing the latter expression of $\boldsymbol{w}$ with the update rule of Relief, Sun and Li [6] concluded that Relief is an online solution to the optimization scheme in Equation (2.15) and they claimed that this is always true except for irrelevant features i.e when $w_i^* = 0$ for $z_i \leq 0$.

2.5.2.2 Iterative Search Margin-Based Algorithm (Simba)

As mentioned in section 2.5.2, the Simba algorithm, proposed by Gilad-Bachrach *et al.* [102], is obtained when using the l_2-norm with this mathematical interpretation. It is an iterative search algorithm that uses a stochastic gradient ascent to maximize its margin-based evaluation function. It is a slightly modified Relief with the major advantage of re-evaluating the margin with respect to the updated weight vector at each previous iteration. It is also said to deal with correlated or redundant features [102].

Definition 2.13. The hypothesis-margin of $\boldsymbol{x}_n$ denoted by $\rho(\boldsymbol{x}_n)$ is given by half the difference between the distance from a data point to its nearmiss and the distance to its nearhit. However, we omit the half [6] obtaining the following:

$$\rho(\boldsymbol{x}_n) = \|\boldsymbol{x}_n - \boldsymbol{M}(\boldsymbol{x}_n)\| - \|\boldsymbol{x}_n - \boldsymbol{H}(\boldsymbol{x}_n)\| \tag{2.23}$$

Where the used distance function $\|\,.\,\|$ is the l_2-norm (or Euclidean distance) defined as:

$$\|\boldsymbol{v}\| = \sqrt{\sum_{i=1}^{F} {v_i}^2} = \sqrt{v_1^2 + ... + v_i^2 + ... + v_F^2} \tag{2.24}$$

[6] In order to maintain the coherence of the hypothesis-margin definition with previous ones, we omit the multiplication by $\frac{1}{2}$ from the definition used in [102] i.e. $\rho(\boldsymbol{x}_n) = \frac{1}{2}\big(\|\boldsymbol{x}_n - \boldsymbol{M}(\boldsymbol{x}_n)\| - \|\boldsymbol{x}_n - \boldsymbol{H}(\boldsymbol{x}_n)\|\big)$.

Definition 2.14. The weighted hypothesis-margin of $\boldsymbol{x}_n$ using the weight vector $\boldsymbol{w} = (w_1,...,w_i,...,w_F)^T$ spanning over F features with respect to $\boldsymbol{X}$ is given by:

$$\rho(\boldsymbol{x}_n,\boldsymbol{w}) = \|\boldsymbol{x}_n - \boldsymbol{M}(\boldsymbol{x}_n)\|_{\boldsymbol{w}} - \|\boldsymbol{x}_n - \boldsymbol{H}(\boldsymbol{x}_n)\|_{\boldsymbol{w}} \tag{2.25}$$

where,

$$\|\boldsymbol{v}\|_{\boldsymbol{w}} = \|\boldsymbol{w} \circ \boldsymbol{v}\| = \sqrt{\sum_{i=1}^{F} w_i{}^2 v_i{}^2} = \sqrt{w_1^2v_1^2 + ... + w_i^2v_i^2 + ... + w_F^2v_F^2} \tag{2.26}$$

Note that, since for any scalar λ, the weighed margin $\rho(\boldsymbol{x}_n,\lambda\boldsymbol{w}) = |\lambda|\rho(\boldsymbol{x}_n,\boldsymbol{w})$, the authors introduced a natural normalization factor, i.e. to require $max(w_i^2) = 1$. This guarantees that $\|\boldsymbol{v}\|_{\boldsymbol{w}} \leq \|\boldsymbol{v}\|$.
The building blocks of Simba's evaluation function are the weighted margins of each data point known to us as $\rho(\boldsymbol{x}_n,\boldsymbol{w})$.
Definition 2.15. Considering the training set $\boldsymbol{X}$ and the feature weights denoted by the vector $\boldsymbol{w}$, Simba's evaluation function is given by:

$$\rho(\boldsymbol{X},\boldsymbol{w}) = \sum_{n=1}^{N} \rho(\boldsymbol{x}_n,\boldsymbol{w}) \tag{2.27}$$

Thus, as $\rho(\boldsymbol{X},\boldsymbol{w})$ is smooth almost everywhere, a gradient ascent can be used to maximize it as follows:

$$\begin{aligned}
\tfrac{\partial\rho(\boldsymbol{X},\boldsymbol{w})}{\partial w_i} &= \textstyle\sum_{n=1}^{N} \tfrac{\partial}{\partial w_i}(\|\boldsymbol{x}_n - \boldsymbol{M}(\boldsymbol{x}_n)\|_{\boldsymbol{w}} - \|\boldsymbol{x}_n - \boldsymbol{H}(\boldsymbol{x}_n)\|_{\boldsymbol{w}}) \\
&= \textstyle\sum_{n=1}^{N} \tfrac{\partial}{\partial w_i}(\|\boldsymbol{q}\|_{\boldsymbol{w}} - \|\boldsymbol{v}\|_{\boldsymbol{w}}) \\
&= \textstyle\sum_{n=1}^{N} \tfrac{\partial}{\partial w_i}\left(\sqrt{\sum_{i=1}^{F} w_i^2q_i^2}\right) - \tfrac{\partial}{\partial w_i}\left(\sqrt{\sum_{i=1}^{F} w_i^2v_i^2}\right) \\
&= \textstyle\sum_{n=1}^{N} \left(\frac{2\ w_iq_i^2}{2\sqrt{\sum_{i=1}^{F} w_i^2q_i^2}}\right) - \left(\frac{2\ w_iv_i^2}{2\sqrt{\sum_{i=1}^{F} w_i^2v_i^2}}\right) \\
&= \textstyle\sum_{n=1}^{N} w_i\left(\frac{q_i^2}{\sqrt{\sum_{i=1}^{F} w_i^2q_i^2}} - \frac{v_i^2}{\sqrt{\sum_{i=1}^{F} w_i^2v_i^2}}\right) \\
&= \textstyle\sum_{n=1}^{N} \left[\frac{(x_{ni}-M(x_n)_i)^2}{\|\boldsymbol{x}_n-\boldsymbol{M}(\boldsymbol{x}_n)\|_{\boldsymbol{w}}} - \frac{(x_{ni}-H(x_n)_i)^2}{\|\boldsymbol{x}_n-\boldsymbol{H}(\boldsymbol{x}_n)\|_{\boldsymbol{w}}}\right] w_i
\end{aligned} \tag{2.28}$$

Where $M(x_n)_i$ and $H(x_n)_i$ denote the values of the nearmiss of $\boldsymbol{x}_n$ and the nearhit of $\boldsymbol{x}_n$ on a specific feature $\boldsymbol{A}_i$ respectively.

Then, Simba algorithm presented in **Algorithm 2.3** uses the following weight updating equation:

$$w_i^{new} = w_i^{old} + \left[\frac{(x_{ni} - M(x_n)_i)^2}{\|\boldsymbol{x}_n - \boldsymbol{M}(\boldsymbol{x}_n)\|_{\boldsymbol{w}}} - \frac{(x_{ni} - H(x_n)_i)^2}{\|\boldsymbol{x}_n - \boldsymbol{H}(\boldsymbol{x}_n)\|_{\boldsymbol{w}}} \right] w_i^{old} \tag{2.29}$$

Algorithm 2.3. Simba

Input:

- Labeled Training set $\boldsymbol{X}$ of N data points and F features
- Number of iterations T

Output: Weight vector $\boldsymbol{w}$

1. Initialize $\boldsymbol{w} = (1, 1, ..., 1)$
2. For $t = 1, ..., T$
   (a) Pick randomly a data point $\boldsymbol{x}$ from $\mathbf{X}$
   (b) Calculate $\boldsymbol{M}(\boldsymbol{x}_n)$ and $\boldsymbol{H}(\boldsymbol{x}_n)$ with respect to $\boldsymbol{X} \backslash \{\boldsymbol{x}_n\}$and the weight vector $\boldsymbol{w}$
   (c) For $i = 1, ..., F$ calculate

$$\Delta_i = \left(\frac{(\Delta(\boldsymbol{A}_i, \boldsymbol{x}_n, \boldsymbol{M}(\boldsymbol{x}_n)))^2}{\Delta_{\boldsymbol{w}}(\boldsymbol{x}_n, \boldsymbol{M}(\boldsymbol{x}_n))} - \frac{(\Delta(\boldsymbol{A}_i, \boldsymbol{x}_n, \boldsymbol{H}(\boldsymbol{x}_n)))^2}{\Delta_{\boldsymbol{w}}(\boldsymbol{x}_n, \boldsymbol{H}(\boldsymbol{x}_n))} \right) w_i$$

   End For
   (d) $\boldsymbol{w} = \boldsymbol{w} + \boldsymbol{\Delta}$

   End For
3. $\boldsymbol{w} \leftarrow \boldsymbol{w}^2 / \left\|\boldsymbol{w}^2\right\|_\infty$
   where, $\boldsymbol{w}^2 = ({w_1}^2, ..., {w_i}^2, ..., {w_F}^2)$ and $\left\|\boldsymbol{w}^2\right\|_\infty = \max({w_1}^2, ..., {w_i}^2, ..., {w_F}^2)$

Although Gilad-Bachrach *et al.* claimed that Simba is superior to Relief as it evaluates distances between data points according to the weight vector $\boldsymbol{w}$ at each iteration, Sun and Li [50], showed that is inferior to ReliefF and highlighted that a major drawback of Simba is its implementation. In fact, its optimization returns many local maxima. Although Simba alleviated this problem by restarting the algorithm multiple times from different starting points, it still cannot guarantee obtaining a global maximum. Moreover, as a constrained non-linear optimization problem, Simba cannot

be easily solved through long-established optimization techniques. Thus, it optimizes its margin-based objective function first through a gradient ascent considering the normalization constraints. Then, as a separate final step, the obtained solution is projected over these constraints as can be seen in step (3) of Simba algorithm. In conclusion, the convergence of Simba's algorithm is doubtful.

2.5.2.3 I-Relief algorithm

After highlighting the limitations of Simba, Sun and Li [6] suggested another iterative Relief-based feature weighting algorithm called I-Relief. In fact, this allowed them to identify two weaknesses in Relief: (1) Its assumption that the nearest neighbors of a data point in the original space are the same in the weighted space (similarly noticed by [108]), and (2) its lacking a mechanism for data outliers detection. Therefore, they proposed an analytical solution to deal with these two issues, hence, describing Relief as an online algorithm that utilizes the performance of a highly non-linear classifier to evaluate a margin-based objective function and solves a simple convex optimization problem in a closed-form.

To handle the two mentioned drawbacks of Relief simultaneously, I-Relief first defines two sets $\mathcal{M}_n = \{m : 1 \leq m \leq N, y_m \neq y_n\}$ and $\mathcal{H}_n = \{m : 1 \leq m \leq N, y_m = y_n, m \neq n\}$ corresponding to each point x_n. Supposing that the nearest hit and nearest miss of each data point are known, their indices are saved in a set denoted $\mathfrak{S}_n = \{(\mathfrak{s}_{n1}, \mathfrak{s}_{n2})\}$ where $\mathfrak{s}_{n1} \in \mathcal{M}_n$ and $\mathfrak{s}_{n2} \in \mathcal{H}_n$. Also, a vector of outliers[7] denoted by $\boldsymbol{o} = [o_1, ..., o_N]^T$ is considered as a set of binary parameters, such that $o_n = 0$ if $\boldsymbol{x}_n$ is an outlier, and $o_n = 1$ otherwise. The objective function in its deterministic form to be optimized using the above-explained Lagrangian is now given by:

$$\rho(w) = \sum_{n=1, o_n=1}^{N} (\|\boldsymbol{x}_n - \boldsymbol{x}_{\mathfrak{s}_{n1}}\|_w - \|\boldsymbol{x}_n - \boldsymbol{x}_{\mathfrak{s}_{n2}}\|_w) \tag{2.30}$$

Since the index sets and $\boldsymbol{o}$ are unknown, their values were assumed to be random variables and the probability distributions of the unobserved data can be derived. Thus, the probability of the m-th data point being the nearmiss of $\boldsymbol{x}_n$ can be defined as:

$$P_{miss}(m|\boldsymbol{x}_n, \boldsymbol{w}) = \frac{f(\|(\boldsymbol{x}_n - \boldsymbol{x}_m)\|_w)}{\sum_{j \in \mathcal{M}_n} f(\|(\boldsymbol{x}_n - \boldsymbol{x}_j)\|_w)} \tag{2.31}$$

[7] An outlier, in our context, is either a mislabeled data point or a one highly corrupted with noise.

Similarly the probability of the m-th data point being the nearhit of $\boldsymbol{x}_n$ is given by:

$$P_{hit}(m|\boldsymbol{x}_n, \boldsymbol{w}) = \frac{f(\|(\boldsymbol{x}_n - \boldsymbol{x}_m)\|_w)}{\sum_{j\in\mathcal{H}_n} f(\|(x_n - \boldsymbol{x}_j)\|_w)} \tag{2.32}$$

Where $f(.)$ is a kernel function that can be denoted as $f(d) = exp(-d/\sigma)$, and the kernel width σ is a user defined parameter.

Likewise the probability of an outlier is defined as:

$$P_o(o_n = 0|\boldsymbol{X}, \boldsymbol{w}) = \frac{\sum_{j\in\mathcal{M}_n} f(\|\boldsymbol{x}_n - \boldsymbol{x}_j\|_w)}{\sum_{\boldsymbol{x}_j\in\boldsymbol{X}\setminus\boldsymbol{x}_n} f(\|(\boldsymbol{x}_n - \boldsymbol{x}_j)\|_w)} \tag{2.33}$$

And the probability of not an outlier is defined as:

$$\gamma_n = 1 - P_o(o_n = 0|\boldsymbol{X}, \boldsymbol{w}^{(t)}). \tag{2.34}$$

where t denotes an iteration and $\boldsymbol{w}^{(t)}$ denotes the value of $\boldsymbol{w}$ at a specific iteration t.

Therefore, upon the latter definitions, the iterative algorithm I-Relief was derived by adopting the idea of the EM algorithm [112] in order to treat unobserved data as random variables, however, the objective function of Equation (2.35) is not a likelihood.
Thus, changing the deterministic $\rho(w)$ into its estimated form as follows:

Step-1: After the t-th iteration, the Q function is calculated as:

$$\begin{aligned}
&Q(\boldsymbol{w}|\boldsymbol{w}^{(t)}) = E_{\{\mathfrak{S},o\}}[\rho(w)] \\
&= \sum_{n=1}^{N} \gamma_n (\sum_{m\in\mathcal{M}_n} \alpha_{m,n} \|\boldsymbol{x}_n - \boldsymbol{x}_m\|_w - \sum_{m\in\mathcal{H}_n} \beta_{m,n} \|\boldsymbol{x}_n - \boldsymbol{x}_m\|_w) \\
&= \sum_{n=1}^{N} \gamma_n \left(\sum_i w_i \sum_{m\in\mathcal{M}_n} \alpha_{m,n} M^i_{n,m} - \sum_i w_i \sum_{m\in\mathcal{H}_n} \beta_{m,n} H^i_{n,m} \right) \\
&= \boldsymbol{w}^T \sum_{n=1}^{N} \gamma_n (\bar{M}_n - \bar{H}_n) \\
&= \boldsymbol{w}^T \boldsymbol{v}
\end{aligned} \tag{2.35}$$

Where superscript i denotes the i-th feature, $M_{n,m} = |\boldsymbol{x}_n - \boldsymbol{x}_m|$ if $m \in \mathcal{M}_n$, $H_{n,m} = |\boldsymbol{x}_n - \boldsymbol{x}_m|$ if $m \in \mathcal{H}_n$, $\alpha_{m,n} = P_{miss}(m|\boldsymbol{x}_n, \boldsymbol{w}^{(t)})$, and $\beta_{m,n} = P_{hit}(m|\boldsymbol{x}_n, \boldsymbol{w}^{(t)})$.
Step 2 : the re-estimation of $\boldsymbol{w}$ in the $(t+1)$-th iteration is :

$$\boldsymbol{w}^{(t+1)} = \arg\max_{\boldsymbol{w}\in\mathcal{W}} Q(\boldsymbol{w}|\boldsymbol{w}^{(t)}) = (\boldsymbol{v})^+ / \|(\boldsymbol{v})^+\|_2 \tag{2.36}$$

Where, $\mathcal{W} = \{\|\boldsymbol{w}\|_2 = 1, \boldsymbol{w} \geq \boldsymbol{0}\}$. Eventually, I-Relief is presented in **Algorithm 2.4**.

Algorithm 2.4. I-Relief

Input:

- Labeled Training set $\boldsymbol{X}$ of N data points and F features
- Number of iterations T
- Kernel width σ , and stopping criterion ϕ

Output: Converged closed-form solution of the weight vector $\boldsymbol{w}$

1. Initialize $\boldsymbol{w} = (1/F, 1/F, ..., 1/F)$
2. For $t = 1, ..., T$
   (a) Calculate pairwise distances $d(\boldsymbol{x}_n, \boldsymbol{x}_m | \boldsymbol{w}^{(t-1)}), \forall \boldsymbol{x}_n, \boldsymbol{x}_m \in \boldsymbol{X}$
   (b) Calculate P_{miss}, P_{hit} and P_o as in (2.31), (2.32) and (2.33) respectively.
   (c) Update the weights in $\boldsymbol{w}$ as in Equation (2.36)
   (d) If $\|\boldsymbol{w}^{(t)} - \boldsymbol{w}^{(t-1)}\| \leq \phi$, **Break** , End For

   End For

2.5.2.4 Extensions of I-Relief to Multi-class and Semi-supervised Contexts

In the same work, I-Relief algorithm was also extended to a handle multi-class problems in two different versions. The first one, called I-Relief-1, combined the mathematical interpretation with ReliefF's way of handling multi-class problems using the below straightforward integration of $\rho = \sum_{c \neq class(\boldsymbol{x})} \frac{P(c)}{1 - P(class(\boldsymbol{x}))} |\boldsymbol{x} - \boldsymbol{M}(\boldsymbol{x}, c)| - |\boldsymbol{x} - \boldsymbol{H}(\boldsymbol{x})|$ into Equation (2.15), where $P(c)$ is the prior probability of class c and $1 - P(class(\boldsymbol{x}))$ represents the sum of probabilities for the misses' classes. However, it was noted that obtaining a positive margin from such a combination does not necessarily mean that a correct classification occurred. Therefore, I-Relief-2 was suggested to regain this characteristic using the below-defined margin:

$$\begin{aligned} \rho &= \min_{\{c \neq class(\boldsymbol{x})\}} |\boldsymbol{x} - \boldsymbol{M}(\boldsymbol{x}, c)| - |\boldsymbol{x} - \boldsymbol{H}(\boldsymbol{x})| \\ &= \min_{\{\boldsymbol{x}_m \in \boldsymbol{X} \setminus \boldsymbol{X}_{class(x)}\}} |\boldsymbol{x} - \boldsymbol{x}_m| - |\boldsymbol{x} - \boldsymbol{H}(\boldsymbol{x})| \end{aligned} \tag{2.37}$$

where $\boldsymbol{X}_{class(x)}$ is a subset of $\boldsymbol{X}$ containing all data points belonging to the class of $\boldsymbol{x}$.

In addition, multiple other extensions were proposed to I-Relief for different goals. One extension, known as online I-Relief, was suggested to work in cases where the amount of training data is very large or when not all of it is initially available [6, 50]. Thus, online I-Relief is a transformation from a batch learning method to an online learning one that is considered computationally more attractive. Another extension, a local learning-based I-Relief algorithm, called Logistic I-Relief, was proposed for cases where we have extremely large data dimensionality with a huge number of irrelevant features. Its main idea is to utilize local learning to break down a complex non-linear problem into a set of locally linear ones and still find the feature weights globally under the same large margin framework.

On the other side, the first extension of RBA's to a semi-supervised context, later known as SLIR (Semi-Supervised Logistic I-Relief), was proposed by Cheng *et al.* [113] as a natural augmentation to the Logistic I-Relief algorithm. Its key idea is in the modification of the objective function of Logistic I-Relief to consider the margins of unlabeled data points as well as the margins of labeled ones. The latter is used to maximize the distance between data points belonging to different classes, while the former is used to extract the geometric structure of the data space. However, SLIR was only capable of handling binary-class problems. Consequently, in a recent work [114], SLIR was extended to deal with multi-class problems in the presence of both labeled and unlabeled data points while ensuring symmetry between them. This algorithm, known as MSLIR (Multi-classification Semi-Supervised Logistic I-Relief), proposes a novel scheme to find the margins of unlabeled data points, by which, it calculates multiple margins for an unlabeled point, each of them corresponds to considering that the point belongs to one of the possible classes. This provides c candidate margin vectors for each unlabeled data point and the chosen margin is the one that maximizes $\boldsymbol{w}$. Finally, Tang and Zhang [115] recorded some poor performance by MSLIR when predicting the labels of unlabeled points and proposed a novel multi-class semi-supervised logistic I-Relief based on nearest neighbor (called MSLIR-NN) to solve this problem. For instance, MSLIR-NN uses the nearest neighbor scheme to label unlabeled data points and then calculates their margin vectors accordingly.

2.5.3 Instance Weighting in RBAs

Knowing that the terms "instance weighting" or "observation weighting", in our context, mean assigning a weight also to each data point contributing in the margin calculation process [2], the majority of initial Relief-based algorithms (from the original Relief to ReliefF algorithm in Table 2.2) did not consider weighting data points. How-

ever, under the umbrella of instance weighting, this can be explained as using binary weights where nearest points contribute with a weight of 1 and all other points get a weight of 0 (i.e no contribution).

For classification problems, Draper *et al.* [108] suggested the first iterative Relief-based approach, called Iterative Relief, with the aim of removing the bias of ReliefF against non-monotonic features[8]. After claiming that ReliefF's distance measure can turn to be irrelevant when dealing with non-monotonic features, it was normal to assign higher weights to monotonic features as they are considered more relevant with respect to non-monotonic ones. Thus, in order to find a distance measure that can effectively choose **similar** nearhits and misses in the feature space, they proposed instance weighting using a "radius", by which, all data points within are included in the calculation of feature weights instead of using an integer number K of nearhits and nearmisses. Moreover, the contribution of a data point decreases as it moves farther from the target point within the radius circle.

Also, the methods suggested by I-Relief algorithms for calculating probability sets for all hits and misses implicitly replaces the whole idea of a specific number K of hits and misses by instance weighting [6, 50, 94, 117]. Where, in fact, a data point with a higher probability of being the nearmiss or nearhit is the one having the smallest distance from the target point.

On the other hand, similarly to I-Relief and its successors, a set of variants that had their main focus on instance weighting were suggested. These are explained thoroughly by the review of Urbanowiz et. al [2] and known as SURF (Spatially Uniform ReliefF), SURF* (* indicates opposite), SWRF* (Sigmoid Weighted ReliefF Star), MultiSURF* (Multiple Threshold SURF*), and MultiSURF (Multiple Threshold SURF). The latter five algorithms could only handle discrete-valued features and binary class problems since they are application driven[9].

[8]The bias of ReliefF was declared by (Bins and Draper, 2001) especially when the data has more noisy features [116], Knowing that a non-monotonic feature is a feature that is neither non-decreasing nor non-increasing with respect to the predicted value. These features can position the data points of the same class into different parts of the feature space even when they are relevant. In other words, it has an internal optima.

[9]These algorithms were suggested for feature selection in genomics problems with single nucleotide polymorphisms (SNPs). In such an application, features can have one of three discrete values (0, 1, or 2) and a binary class representing sick or healthy patients.

2.6 Statistical Interpretation

As mentioned before, RBAs are non-parametric from the statistical point of view, which means, they do not make any assumption about the underlying probability distribution of data. This makes determining the statistical significance of feature weights assigned by RBAs a hard task. However, Le *et al.* [7] re-conceptualized Relief to obtain a new family of STatistical Inference Relief (STIR) estimators that maintains Relief's ability to detect feature interactions while integrating sample variance of the nearest neighbor distances into the feature relevancy estimation. They claimed that the feature weights obtained by the standard Relief algorithm are equivalent to a difference of mean feature value differences between nearhit and nearmiss groups. Accordingly, a new formulation of Relief estimates that accounts for the variance within and between groups is suggested. The STIR formalism applies generally to all RBAs, including ReliefF with fixed K-nearest neighbors and MultiSURF with adaptive radii. Thus, a set of definitions was derived as follows:

Definition 2.16. The set of ordered nearmiss pairs denoted by $(\boldsymbol{x}_n, \boldsymbol{M}_{j_n}(\boldsymbol{x}_n))$, or simply, $(\boldsymbol{x}_n, \boldsymbol{M}_{j_n})$ of T data points $(n = 1, ..., T)$ and nearest K_{M_n} misses, denoted $M_{j_n} (j = 1, ..., K)$, are represented as the following nested set:

$$M = \{\{(\boldsymbol{x}_n, \boldsymbol{M}_{j_n})\}, \quad j_n = 1, ..., K_{M_n};\ n = 1, ..., T\} \tag{2.38}$$

Definition 2.17. The set of ordered nearhit pairs denoted by $(\boldsymbol{x}_n, \boldsymbol{H}_{j_n})$ of T data points $(n = 1, ..., T)$ nearest K_{H_n} hits, denoted $H_{j_n} (j = 1, ..., K)$, are represented as the following nested set:

$$H = \{\{(\boldsymbol{x}_n, \boldsymbol{H}_{j_n})\}, \quad j_n = 1, ..., K_{H_n};\ n = 1, ..., T\} \tag{2.39}$$

Where for both hit and miss sets, the inner index j_n depends on the outer index n. The importance of this index appears in algorithms like MultiSURF where each data point $\boldsymbol{x}_n$ can have a different number of misses and hits instead of a fixed number K.

Now that the sets are clearly defined, we show the reformulation of the weights assigned by RBAs into the difference of the means of these hit and miss sets. First, note that, the hit and miss sets in Equations (2.39) and (2.38) respectively are calculated using the distance $\Delta(\boldsymbol{p}_1, \boldsymbol{p}_2)$ of Equation (2.6), taking into consideration the "neighborhood" definition (e.g fixed K in ReliefF and data radius in MultiSURF).

Definition 2.18. The mean of Δ function (defined in Equation (2.1)) of a feature $\boldsymbol{A}_i$ averaged over all pairs of nearmisses (M Equation (2.38)), denoted $\bar{M}_i$, can be expressed as:

$$\bar{M}_i = \frac{1}{T}\sum_{n=1}^{T}\frac{1}{K_{M_n}}\sum_{j_n=1}^{K_{M_n}}\Delta(\boldsymbol{A}_i, \boldsymbol{x}_n, \boldsymbol{M}_{j_n}) \tag{2.40}$$

where M_{j_n} is the j-th nearest miss of the data point $\boldsymbol{x}_n$, and K_{M_n} is the number of nearmisses of $\boldsymbol{x}_n$. The scaling by $\frac{1}{K_{M_n}}$ ensures a consistent neighborhood average weighting for different RBAs like MultiSURF, SURF and ReliefF.

Definition 2.19. The mean of Δ function (defined in Equation (2.1)) of a feature $\boldsymbol{A}_i$ averaged over all pairs of nearhits (H Equation (2.39)), denoted $\bar{H}_i$, can be expressed as:

$$\bar{H}_i = \frac{1}{T}\sum_{n=1}^{T}\frac{1}{K_{H_n}}\sum_{j_n=1}^{K_{H_n}}\Delta(\boldsymbol{A}_i, \boldsymbol{x}_n, \boldsymbol{H}_{j_n}) \tag{2.41}$$

where, similarly, H_{j_n} is the j-th nearest hit of the data point $\boldsymbol{x}_n$, and K_{H_n} is the number of nearhits of $\boldsymbol{x}_n$.

Definition 2.20. The importance score assigned by RBAs to each feature $\boldsymbol{A}_i$ can be then expressed as the averaged z presented in Equation (2.17) as follows:

$$\bar{z_i} = \bar{M}_i - \bar{H}_i \tag{2.42}$$

This formulation applies to all RBAs [7]. Accordingly, Equation (2.42) was used later as the basis to compute the permutation p-values for the sake of comparison. Then, for computing the statistical significance of features with computational efficiency, Equation (2.42) was extended to develop a Relief-based pseudo t-test.

2.6.1 STatistical Inference for Relief (STIR)

The new type of Relief-based algorithm is then presented as an incorporation of the pooled standard deviations about the mean hit and miss Δ functions to transform the Relief-based feature importance scores $\bar{z_i}$ into a pseudo t-statistic [7]. Note that, the pooled standard deviation presented below allows for unbalanced variances between hit and miss nearest neighbor Δ functions and allows for a different number of these Δ functions in the hit and miss groups themselves.

Definition 2.21. The STIR weight (or STIR score) is constructed from the Relief difference of means ($\bar{z_i}$ in Equation (2.42)) in the numerator and the standard error in

the denominator as follows:

$$w_{STIR_i} = \frac{\bar{M}_i - \bar{H}_i}{\sigma_{pool}[M,H]\sqrt{1/|M| + 1/|H|}} \tag{2.43}$$

Where $|H| = \sum_{n=1}^{T} K_{H_n}$ and $|M| = \sum_{n=1}^{T} K_{M_n}$ are the total number of hit and miss neighbors over all the data points in $\boldsymbol{X}$. And the pooled standard deviation denoted by σ_{pool} is given by:

$$\sigma_{pool}[M,H] = \sqrt{\frac{(|M|-1)\sigma^2_{\bar{M}_i} + |H|-1)\sigma^2_{\bar{H}_i}}{|M| + |H| - 2}} \tag{2.44}$$

And the group variances are given by:

$$\sigma^2_{\bar{M}_i} = \frac{1}{T}\sum_{n=1}^{T}\frac{1}{K_{M_n}}\sum_{j_n=1}^{K_{M_n}}(\Delta(\boldsymbol{A}_i, \boldsymbol{x}_n, \boldsymbol{M}_{j_n}) - \bar{M}_i)^2 \tag{2.45}$$

$$\sigma^2_{\bar{H}_i} = \frac{1}{T}\sum_{n=1}^{T}\frac{1}{K_{H_n}}\sum_{j_n=1}^{K_{H_n}}(\Delta(\boldsymbol{A}_i, \boldsymbol{x}_n, \boldsymbol{H}_{j_n}) - \bar{H}_i)^2 \tag{2.46}$$

According to [7], w_{STIR_i} score shown in Equation (2.43) approximately follows a t-distribution from which they compute p-values. For that, a degree of freedom $df = |M| + |H| - 2$ is used in calculating the p-value. It was concluded that STIR can record the statistical significance of RBA scores by a pseudo t-test that stands for variance in the mean difference of hit and miss nearest neighbor Δ functions.

2.7 Conclusion

As we were interested in Relief-based methods, in this chapter, we concisely reviewed the set of well-known efficient filter-type RBAs. For that, we started with a clear explanation of the original Relief algorithm while highlighting its strengths and limitations. Afterward, we presented the main variants and extensions of Relief that were suggested to handle problems with noisy, incomplete, and multi-class data. This was done while generally categorizing RBAs according to their four broad possible interpretations. These are, the probabilistic, the comprehensible, the mathematical and the statistical interpretations. Most importantly, Relief was introduced as a margin-based feature selection algorithm which will be the base of the following chapters.

Table 2.2: Summary of the main Relief-based algorithms covered in this chapter. Each algorithm is followed by its reference, main contribution, learning context, time complexity, and other capabilities mentioned explicitly in their original publications. C, l, and u used in the semi-supervised algorithms denote the number of classes, labeled data points and unlabeled data points respectively. Some parts of the table are adapted from [2].

Algorithm / Reference(s)	Focus / Contribution	Learning Context	Asymptotic Time Complexity	Continuous Features	Instance Weighting	Multi-Class Problems	Regression Problems	Missing Data	Noise-tolerant	Feature-Redundancy	Iterative Approach	Outlier Detection
Relief [66]	First filter-type feature selection method capable of dealing with feature interactions (dependencies) in a simple and efficient way.	Supervised	O(TNF)	x								
ReliefA [4]	Using k-nearest hits/misses instead of only one to increase the reliability of probability approximations and feature weights when data is noisy	Supervised	O (TNF)	x					x			
ReliefB-D [4]	Handling missing data with ReliefD being the best variant	Supervised	O (TNF)	x				x	x			
ReliefE-F [4]	Dealing with multi-class problems with ReliefF becoming the standard algorithm.	Supervised	O (TNF)	x		x		x	x			
RReliefF [99, 101]	Adapting ReliefF to work with continuous class labels. Introducing instance weighting by distance-from-target.	Supervised	O (TNF)	x	x		x	x	x			
Relieved-F [100]	Choosing the nearest points for scoring in a deterministic way. Handling missing data.	Supervised	$T = N \implies$ O(N^2F)	x		x		x	x			
Iterative Relief [108]	Introducing the first iterative approach. Improving ReliefF by removing its bias against non-monotonic features. Replacing k-nearest neighbors by radius-circle around the target point. Using distance-from-target instance weighting.	Supervised	O(TN^2F)	x	x				x		x	

Table 2.2 Continued

Algorithm / Reference(s)	Focus / Contribution	Learning Context	Asymptotic Time Complexity	Continuous Features	Instance Weighting	Multi-Class Problems	Regression Problems	Missing Data	Noise-tolerant	Feature-Redundancy	Iterative Approach	Outlier Detection
Iterative Search Margin-based Algorithm (Simba) [102]	Introducing the first margin concept to Relief-based algorithms. Considering scoring in the weighted space. Using gradient ascent optimization on a margin-based objective function.	Supervised	O (TNF)	x		x				x	x	
I-Relief [6, 50]	Introducing Relief from its optimization perspective. Replacing the idea of defined K-nearest neighbors by probabilities of being a hit or a miss and using them for instance weighting. Introducing outlier detection.	Supervised	O (TN^2F)	x	x				x		x	x
I-Relief-1 and **I-Relief-2** [6, 50]	Extending I-Relief to deal with multi-class problems.	Supervised	O (TN^2F)	x	x	x					x	x
Online I-Relief [6, 50]	Dealing with huge amounts of data or data not completely available initially through online learning (not batch mode).	Supervised	O (TNF)	x	x	x					x	x
Logistic I-Relief (LIR) [94, 117]	Handling extremely large data dimensionality with a large number of irrelevant features in an efficient way through local learning. Introducing a logistic margin-based objective function with a sparse solution.	Supervised	O (TN^2F)	x	x	x			x		x	
Semi-supervised Logistic I-Relief (SLIR) [113]	Modifying the objective function of LIR to consider margins of both labeled and unlabeled data points.	Semi-supervised	Margin of labeled: O(Fl^2). Margin of unlabeled O(Flu)	x	x						x	

Table 2.2 Continued

Algorithm / Reference(s)	Focus / Contribution	Learning Context	Asymptotic Time Complexity	Continuous Features	Instance Weighting	Multi-Class Problems	Regression Problems	Missing Data	Noise-tolerant	Feature-Redundancy	Iterative Approach	Outlier Detection
Multi-classification Semi-supervised Logistic I-Relief (MSLIR)[114]	Modifying the objective function of SLIR to consider margins of both labeled and unlabeled data points in multi-class problems.	Semi-supervised	Margin of labeled: $O(Fl^2)$. Margin of unlabeled $O(CFlu)$	x	x	x					x	
Multi-class Semi-supervised Logistic I-Relief based on nearest neighbor (MSLIR-NN) [115]	Using the nearest neighbor scheme to label unlabeled points and then calculating their margin vectors accordingly.	Semi-supervised	Margin of labeled: $O(Fl^2)$. Margin of unlabeled $O(Flu)$	x	x	x					x	
STatistical Inference Relief (STIR) [7]	Re-conceptualizing Relief to obtain a new family of STatistical Inference Relief (STIR) estimators while maintaining its ability to detect feature interactions through integrating sample variance of the nearest neighbor distances into the feature relevancy estimation.	Supervised	ϕ	x	x				x			

Chapter 3

An Approach based on Hypothesis-Margin and Pairwise Constraints

Contents

3.1 Introduction

The concept of a hypothesis-margin was initially suggested in the supervised learning context by [66] as an algorithm called Relief, it is a filter-type method for individual-evaluation feature selection. Being aware of the contextual supervision information and capable of calculating a weight value for each feature, Relief proved its ability to successfully rank features in the order of their relevance to a specific problem like classifying new data points. This importance measure is referred to as feature weights. By meaning, the hypothesis-margin is the largest distance a data point can travel within the feature space without altering the dataset's labeling structure, and thus without affecting the label prediction of a new arriving point.

However, as explained in Chapter 2, in many real-world applications, there usually exist few labeled data points and lots of unlabeled ones, by which both supervised and unsupervised feature selection algorithms cannot fully take advantage of all data points in this scenario. Thus, it was wise to use semi-supervised methods [39, 54] to feat both labeled and unlabeled data points. In fact, compared to class labels, pairwise constraints are another type of supervision information that can be acquired more easily. These constraints simply specify whether two data points are similar and therefore should be in the same category (a must-link constraint) or dissimilar and therefore should be in different categories (a cannot-link constraint) without the need to identify their class labels.

In this chapter, we suggest to integrate the modification of hypothesis-margin when used with cannot-link constraints, with the analytical solution of the supervised Relief algorithm [50, 66]. This means we conjugate hypothesis-margin concept used in [107] with Relief algorithm from its optimization perspective. The proposed method is a semi-supervised margin-based feature selection algorithm called **Relief-Sc**. It is a modification of the well-known original Relief algorithm from its optimization perspective. It utilizes cannot-link constraints only to solve a simple convex problem in a closed form giving a unique solution. Besides, a slightly modified Relief-Sc into a more robust version with respect to noisy data, called **ReliefF-Sc** algorithm, is also presented and generally applied instead in our experiments.

In addition, we propose to extend our semi-supervised feature selection method into a novel combination of feature clustering and hypothesis margin maximization called **FCRSC**. This approach aims at handling the two core aspects of feature selection (relevance and redundancy) and is divided into three steps. First, the similarity weights between features are represented by a sparse graph by which it is assumed that each feature can be reconstructed from the sparse linear combination of the others. Second, features are then hierarchically clustered identifying groups of the most similar ones. Finally, our semi-supervised margin-based objective function is optimized to select the most data discriminative feature from within each cluster, hence, maximizing relevance while minimizing redundancy (maximizing diversity) among features. Eventually, we experimentally validate our proposed approach on multiple well-known UCI [3] benchmark datasets.

This chapter is divided into two main sections. The first main one, i.e. section 3.2, provides the introduction and explanation of the notions and concepts of the hypothesis-margin in a semi-supervised learning context with a general mathematical interpretation (section 3.2.1) and three feature selection algorithms descending from it. These are Relief-Sc, ReliefF-Sc and Simba-Sc presented in sections 3.2.2, 3.2.3, and 3.2.4 respectively. The second main section 3.3 presents the feature selection framework that deals with feature redundancy in addition to feature relevance. It includes sections 3.3.1, 3.3.2, and 3.3.3 that present building the sparse graph, hierarchical clustering upon it, and the combination of feature clustering with constrained margin-based feature selection respectively. On the other side, in section 3.4, we present the experimental results to validate the proposed approaches. Experiments are achieved on multiple well-known UCI machine learning datasets [3]. Finally, we conclude in section 3.5.

3.2 Hypothesis-Margin in a Constrained Context For Maximizing Relevance

In this section, we present a concise explanation of the constrained hypothesis-margin under a general mathematical interpretation similarly to that of Chapter 2. We also present the proposed margin-based feature selection algorithm Relief-Sc. In addition, we explain Simba-Sc [107] as a state of the art algorithm that applies feature selection in a way that is similar to ours.

3.2.1 General Mathematical Interpretation

Hypothesis-margin plays an important role in feature selection, just like sample-margin does in Support Vector Machines (SVM) [94]. As hypothesis-margin lower bounds the sample margin [102, 106], it was stated by Yang and Song [107] that in the case of a 1-Nearest Neighbor (1-NN) large hypothesis-margin, a large sample-margin is guaranteed with ease of computation. In cases where the only available supervision information is in the form of pairwise constraints, especially cannot-link ones [107], we consider using these constraints to calculate the constrained hypothesis-margin. Knowing that a cannot-link constraint specifically indicates that each data point of the considered data pair should belong to a different cluster, it is necessary to modify the supervised notions of nearmiss and nearhit in order to find the constrained hypothesis-margin. Thus, for pedagogical purposes, all the following definitions and equations are adapted from Chapter 2.

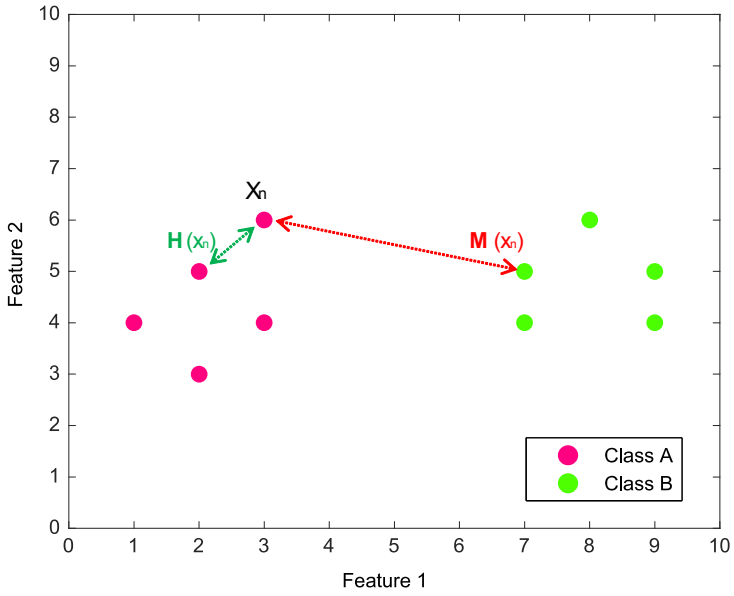
Definition 3.1. Let $CL = \{(\boldsymbol{x}_n, \boldsymbol{x}_m)\}$ be a set of cannot link constraints and $(\boldsymbol{x}_n, \boldsymbol{x}_m)$ be one cannot-link constraint in it. For instance, as can be seen in Figure (3.1), in the constrained context, the nearhit of $\boldsymbol{x}_m$ denoted $\boldsymbol{H}(\boldsymbol{x}_m)$ is considered equivalent to the nearmiss $\boldsymbol{M}(\boldsymbol{x}_n)$ of $\boldsymbol{x}_n$ (that is usually used in a supervised context with class labels), thus, it replaces it. On the other side, the nearhit of $\boldsymbol{H}(\boldsymbol{x}_n)$ represents the nearhit of $\boldsymbol{x}_n$ as it is usually. Also, the number of constraints $card(CL)$ is equal to a user-predefined value T.

Note that, according to the local cluster assumption, the nearhit of a point is simply its nearest neighbor without any class information. In fact, this assumption means that nearby points are assigned to the same class or in other words have the same label. Accordingly, the constrained hypothesis-margin was first mentioned in [107].

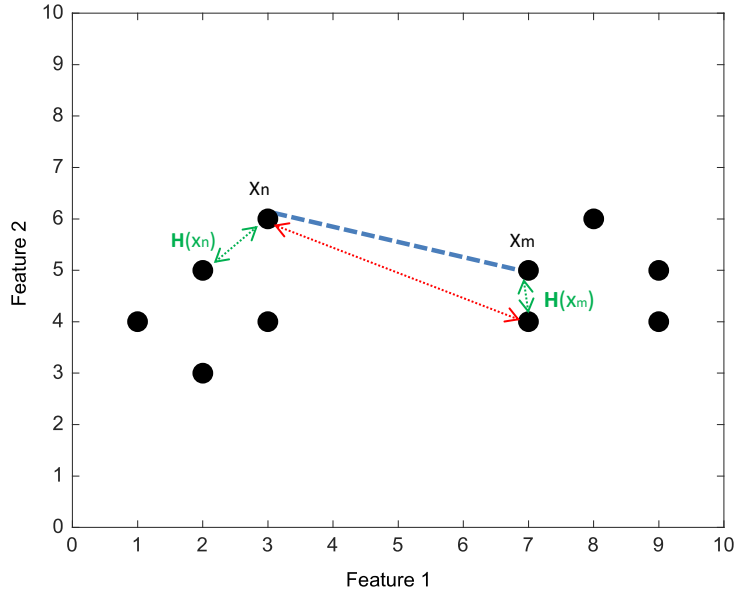
Definition 3.2. The constrained hypothesis-margin is calculated as the difference between the distance from the data point $\boldsymbol{x}_n$ to the nearhit of $\boldsymbol{x}_m$ denoted $\boldsymbol{H}(\boldsymbol{x}_m)$ and the distance from the data point $\boldsymbol{x}_n$ to its own nearhit $\boldsymbol{H}(\boldsymbol{x}_n)$.

$$\rho(\boldsymbol{x}_n, \boldsymbol{x}_m) = \Delta(\boldsymbol{x}_n, \boldsymbol{H}(x_m)) - \Delta(\boldsymbol{x}_n, \boldsymbol{H}(x_n)) \tag{3.1}$$

recalling that $\Delta(\boldsymbol{p}_1, \boldsymbol{p}_2)$ is defined in Chapter 2 as the generalized distance function between any two data points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ over all the features F with $p \in \mathbb{R}$ and $\Delta(\boldsymbol{A}_i, \boldsymbol{p}_1, \boldsymbol{p}_2)$ defined in **Definition 2.3**.



(a) The original notions of nearmiss and nearhit used in the supervised margin calculation (see Figure 2.1a). The nearmiss of a point $\boldsymbol{x}_n$ represented by $\boldsymbol{M}(\boldsymbol{x}_n)$ is its nearest point from a different class and the nearhit of a point $\boldsymbol{x}_n$ represented by $\boldsymbol{H}(\boldsymbol{x}_n)$ is its nearest point from the same class.



(b) The modified notions of nearmiss and nearhit used in the constrained margin calculation, where the dashed line represents the cannot-link constraint between a couple ($\boldsymbol{x}_n$,$\boldsymbol{x}_m$). $\boldsymbol{H}(\boldsymbol{x}_m)$ represents the nearmiss $\boldsymbol{M}(\boldsymbol{x}_n)$ of $\boldsymbol{x}_n$ and $\boldsymbol{H}(\boldsymbol{x}_n)$ represents the nearhit of $\boldsymbol{x}_n$.

Figure 3.1: The evolution of the nearmiss and nearhit notions from the supervised context to the semi-supervised constrained one.

Therefore, the overall constrained hypothesis-margin with respect to the set of cannot-link constraints CL and the training set $\boldsymbol{X}$ can be computed as:

$$\rho(\boldsymbol{X}) = \sum_{(\boldsymbol{x}_n, \boldsymbol{x}_m) \in CL} \rho(\boldsymbol{x}_n, \boldsymbol{x}_m) \tag{3.2}$$

Similarly to the supervised context, we consider the weighted hypothesis-margin using the weight vector $\boldsymbol{w}$ defined in **Definition 2.2**, however, in a semi-supervised constrained context. The latter can be calculated as follows:

$$\rho((\boldsymbol{x}_n, \boldsymbol{x}_m), \boldsymbol{w}) = \Delta_{\boldsymbol{w}}(\boldsymbol{x}_n, \boldsymbol{H}(\boldsymbol{x}_m)) - \Delta_{\boldsymbol{w}}(\boldsymbol{x}_n, \boldsymbol{H}(\boldsymbol{x}_n)) \tag{3.3}$$

Each time the margin is calculated with respect to a data point $\boldsymbol{x}_n$ of F features, the weight vector $\boldsymbol{w}$ contributes to this calculation. Taking Equation (3.3) into account, it can be generalized to calculate the weighted hypothesis-margin over all the given data points. Consequently, the final weight vector of length F shows the impact of each feature in enlarging the margin, which is considered the score/weight of the feature.

Then, the weighted constrained hypothesis-margin with respect to the set of cannot-link constraints CL and the training set $\boldsymbol{X}$ can be computed as:

$$\rho(\boldsymbol{X}, \boldsymbol{w}) = \sum_{(\boldsymbol{x}_n, \boldsymbol{x}_m) \in CL} \rho((\boldsymbol{x}_n, \boldsymbol{x}_m), \boldsymbol{w}) \tag{3.4}$$

It is then clear that Equation (3.4) is the summation of the weighted constrained hypothesis margins over the whole cannot-link constraints set. So, the feature subset that contributes the most in enlarging this overall margin has a weight vector of higher values, and is consequently selected.

Similarly to the supervised context, changing the value of superscript p in Equations (2.6) and (2.9) defines different constrained margin-based feature selection algorithms. If the l_1-norm is used, deriving Manhattan distance, our proposed constrained Relief-Sc algorithm is obtained [118]. However, if the l_2-norm is used, deriving Euclidean distance, the resulting algorithm is the constrained Simba-Sc [107].

3.2.2 Relief with Side Constraints (Relief-Sc)

We consider the interesting mathematical interpretation that was suggested by Sun and Li [6] to view the margin from an optimization perspective. It was stated that this interpretation not only takes advantage of the performance of a highly nonlinear clas-

sifier to calculate a margin-based objective function, but also solves a simple convex problem in a closed-form and obtains a unique solution. Thus, we modified this formulation into a constrained one called Relief-Sc that obtains its solution by calculating the margin over all the constraints in CL in a batch mode.

Definition 3.3. The constrained hypothesis-margin of a cannot-link constraint $(\boldsymbol{x}_n, \boldsymbol{x}_m)$ denoted by $\rho(\boldsymbol{x}_n, \boldsymbol{x}_m)$ under Relief-Sc is defined as:

$$\rho(\boldsymbol{x}_n, \boldsymbol{x}_m) = |\boldsymbol{x}_n - \boldsymbol{H}(\boldsymbol{x}_m)| - |\boldsymbol{x}_n - \boldsymbol{H}(\boldsymbol{x}_n)| \tag{3.5}$$

Where the used distance function $|.|$ is the l_1-norm (or Manhattan distance) defined in Equation (2.12).

Accordingly, the weighted constrained hypothesis-margin of the cannot-link constraint $(\boldsymbol{x}_n, \boldsymbol{x}_m)$ with respect to $\boldsymbol{X}$ is:

$$\rho((\boldsymbol{x}_n, \boldsymbol{x}_m), \boldsymbol{w}) = |\boldsymbol{x}_n - \boldsymbol{H}(\boldsymbol{x}_m)|_{\boldsymbol{w}} - |\boldsymbol{x}_n - \boldsymbol{H}(\boldsymbol{x}_n)|_{\boldsymbol{w}} \tag{3.6}$$

where $|\boldsymbol{v}|_{\boldsymbol{w}}$ is defined in Equation (2.14).

Therefore, we focus on the constrained environment using pairwise constraints, especially cannot-link ones as they are considered more important than must-link constraints from margin's perspective [107]. As a result, we found it axiomatic to integrate the usage of constrained hypothesis-margin with a more simple and computationally efficient algorithm like Relief from its optimization point of view. Since constrained algorithms are penalized by the choice of constraints, the unique solution provided by optimized Relief-Sc allows it to be effectively analyzed when given a fixed set of constraints. Thus, the optimization method (gradient ascent) used in [107] was replaced by the analytical solution suggested in [6] producing a new version of Relief algorithm from an optimization view point.

Definition 3.4. The weighted constrained margin-based objective function to be optimized by Relief-Sc is given by:

$$\begin{aligned}
&\max_{\boldsymbol{w}} \sum_{(\boldsymbol{x}_n, \boldsymbol{x}_m) \in CL} \rho((\boldsymbol{x}_n, \boldsymbol{x}_m), \boldsymbol{w}) \\
&= \max_{\boldsymbol{w}} \sum_{(\boldsymbol{x}_n, \boldsymbol{x}_m) \in CL} \left(\sum_{i=1}^{F} w_i |\boldsymbol{x}_{ni} - \boldsymbol{H}(x_m)_i| - \sum_{i=1}^{F} w_i |\boldsymbol{x}_{ni} - \boldsymbol{H}(x_n)_i| \right) \\
&\text{s.t. } \|\boldsymbol{w}\|_2^2 = 1 \text{ and } \boldsymbol{w} \geq \boldsymbol{0}
\end{aligned} \tag{3.7}$$

It is intuitive that the weight vector $\boldsymbol{w} \geq \mathbf{0}$ should have positive values for relevant features since it is a distance metric, and $||\boldsymbol{w}||_2^2 = 1$ prevents the vector from being maximized without bounds [6].

Definition 3.5. To simplify and combine Equations (3.6) and (3.7), we show the i-th element (corresponding to the i-th feature) of the constrained margin vector $\boldsymbol{z}$. It is evaluated over all the available cannot-link constraints in CL with respect to the training set $\boldsymbol{X}$ as follows:

$$z_i = \sum_{(\boldsymbol{x}_n, \boldsymbol{x}_m) \in CL} |x_{ni} - H(\boldsymbol{x}_m)_i| - |x_{ni} - H(\boldsymbol{x}_n)_i| \tag{3.8}$$

For clarity, we express the margin vector $\boldsymbol{z}$ in the following notation:

$$\boldsymbol{z} = \sum_{(\boldsymbol{x}_n, \boldsymbol{x}_m) \in CL} \left(\begin{pmatrix} |x_{n1} - H(x_m)_1| \\ \dots \\ |x_{nF} - H(x_m)_F| \end{pmatrix} - \begin{pmatrix} |x_{n1} - H(x_n)_1| \\ \dots \\ |x_{nF} - H(x_n)_F| \end{pmatrix} \right) = \begin{pmatrix} z_1 \\ \dots \\ z_F \end{pmatrix} \tag{3.9}$$

Definition 3.6. Considering the constrained margin vector $\boldsymbol{z}$, the simplified weighted constrained margin-based objective function to be optimized by Relief-Sc can be formulated as follows:

$$\max_{\boldsymbol{w}} \quad \boldsymbol{w}^T \boldsymbol{z}, \quad \text{s.t.} \quad \|\boldsymbol{w}\|_2^2 = 1, \quad \boldsymbol{w} \geq \mathbf{0} \tag{3.10}$$

Now, using the Lagrangian function the Equation (3.10) can be expressed as:

$$\mathcal{L}(\boldsymbol{w}, \lambda, \boldsymbol{\mu}) = -\boldsymbol{w}^T \boldsymbol{z} + \lambda(\boldsymbol{w}^T \boldsymbol{w} - 1) - \boldsymbol{\mu}^T \boldsymbol{w} \tag{3.11}$$

where $\lambda > 0$ is the Lagrange multiplier and $\boldsymbol{\mu} \geq \mathbf{0}$ is the Karush Kuhn Tucker (KKT) multiplier vector satisfying complementarity condition: $\boldsymbol{\mu}^T \boldsymbol{w} = 0$.

Now by calculating the derivative of $\mathcal{L}$ with respect to $\boldsymbol{w}$ we obtain:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = -\boldsymbol{z} + 2\lambda \boldsymbol{w} - \boldsymbol{\mu} \tag{3.12}$$

and we satisfy the stationarity condition by solving $\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = 0$ to obtain $\boldsymbol{w}$ as follows:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = 0 \implies \boldsymbol{w} = \frac{1}{2\lambda}(\boldsymbol{z} + \boldsymbol{\mu}) \tag{3.13}$$

As mentioned in section 2.5.2.1, Sun and Li [6] used the Karush Kuhn Tucker (KKT [111]) complementarity condition: $\boldsymbol{\mu}^T\boldsymbol{w} = 0$ to verify the three following cases that also hold in Relief-Sc:

1. $z_i = 0 \Rightarrow \mu_i = 0,\ w_i = 0.$

2. $z_i > 0 \Rightarrow z_i + \mu_i > 0 \Rightarrow w_i > 0 \Rightarrow \mu_i = 0$

3. $z_i < 0 \Rightarrow \mu_i > 0 \Rightarrow w_i = 0 \Rightarrow z_i = -\mu_i$

This leads to:

$$\left\{w_i\right\}_{i=1}^{F} = \begin{cases} \frac{{z_i}^+}{2\lambda} & \text{if } z_i > 0 \\ 0 & \text{if } z_i \leq 0 \end{cases} \tag{3.14}$$

Finally, the optimal solution can be calculated in a closed-form as:

$$\begin{aligned} &\boldsymbol{w}^* = \boldsymbol{w}/\|\boldsymbol{w}\|_2 = (\boldsymbol{z})^+/\|(\boldsymbol{z})^+\|_2 \\ &\text{where, } \boldsymbol{z} \text{ is given by Equation } (3.9) \\ &\text{and } (\boldsymbol{z})^+ = [\max(z_1,0),...,\max(z_F,0)]^T \end{aligned} \tag{3.15}$$

In summary, Relief-Sc uses the cannot-link constraint set CL to calculate the averaged margin $\boldsymbol{z}$ that the features induce using the nearhits of $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$. Finally, it calculates $\boldsymbol{w}$ directly as shown in step 3 of **Algorithm 3.1**.

Algorithm 3.1 Relief-Sc

Input: Training data $\boldsymbol{X}$, Set of cannot-link constraints CL
Output: Weight vector $\boldsymbol{w}$

1. Calculate $\boldsymbol{H}(\boldsymbol{x}_n)$ and $\boldsymbol{H}(\boldsymbol{x}_m)$ for each cannot-link constraint in CL with Respect to $\boldsymbol{X}$

2. For $i = 1,...,F$,
   $z_i = \sum_{(\boldsymbol{x}_n,\boldsymbol{x}_m)\in CL} \Delta(\boldsymbol{A}_i,\boldsymbol{x}_n,\boldsymbol{H}(\boldsymbol{x}_m)) - \Delta(\boldsymbol{A}_i,\boldsymbol{x}_n,\boldsymbol{H}(\boldsymbol{x}_n))$
   end For

3. $\boldsymbol{w} = (\boldsymbol{z})^+/||(\boldsymbol{z})^+||_2$
   where, $(\boldsymbol{z})^+ = [max(z_1,0),...,max(z_F,0)]^T$

3.2.3 ReliefF-Sc: A Robust version of Relief-Sc

However, in ReliefF [4], the hypothesis-margin is calculated over a group of K-nearest neighbors (K-NN) for each selected data point which leads to considering ReliefF as the robust supervised version of the original Relief algorithm with respect to noisy data. Thus, we similarly enhance Relief-Sc. For instance, when considering a cannot-link constraint $(\boldsymbol{x}_n, \boldsymbol{x}_m)$, instead of taking only one nearhit to $\boldsymbol{x}_n$ and one nearhit to $\boldsymbol{x}_m$, we now consider K-nearhits to $\boldsymbol{x}_n$ and K-nearhits to $\boldsymbol{x}_m$ denoted by $K\boldsymbol{H}(\boldsymbol{x}_n) : \{\boldsymbol{H}_1(\boldsymbol{x}_n), \boldsymbol{H}_2(\boldsymbol{x}_n), ..., \boldsymbol{H}_K(\boldsymbol{x}_n)\}$ and $K\boldsymbol{H}(\boldsymbol{x}_m) : \{\boldsymbol{H}_1(\boldsymbol{x}_m), \boldsymbol{H}_2(\boldsymbol{x}_m), ..., \boldsymbol{H}_K(\boldsymbol{x}_m)\}$ respectively, where K is a user defined parameter representing the number of closest points to $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$ to be considered. In this way, the margin evaluation is averaged over a larger neighborhood and thus is less vulnerable to erroneous data. Consequently, the i-th element (corresponding to the i-th feature) of the margin vector $\boldsymbol{z}$ is given by:

$$z_i = \sum_{(\boldsymbol{x}_n, \boldsymbol{x}_m) \in CL} \frac{1}{K} \sum_{j=1}^{K} \left(\Delta(\boldsymbol{A}_i, \boldsymbol{x}_n, \boldsymbol{H}_j(\boldsymbol{x}_m)) - \Delta(\boldsymbol{A}_i, \boldsymbol{x}_n, \boldsymbol{H}_j(\boldsymbol{x}_n)) \right) \tag{3.16}$$

Therefore, in the context of a robust constrained hypothesis margin over K-NN, we apply Relief-Sc as stated in **Algorithm 3.2**. To sum up, it uses a cannot-link constraint set to calculate the average margin z_i that each feature induces over the K-nearest hits of $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$. Finally, it calculates $\boldsymbol{w}$ directly as shown in step 3 of **Algorithm 3.2**.

Algorithm 3.2. ReliefF-Sc

Input: Training data $\boldsymbol{X}$, Set of cannot-link constraints CL, and Number of Nearest Neighbors K

Output: Weight vector $\boldsymbol{w}$

1. Calculate $K\boldsymbol{H}(\boldsymbol{x}_n)$ and $K\boldsymbol{H}(\boldsymbol{x}_m)$ for each cannot-link constraint in CL with Respect to $\boldsymbol{X}$
2. For $i = 1, ..., F$,
   $z_i = \sum_{(\boldsymbol{x}_n, \boldsymbol{x}_m) \in CL} \frac{1}{K} \sum_{j=1}^{K} \left(\Delta(\boldsymbol{A}_i, \boldsymbol{x}_n, \boldsymbol{H}_j(\boldsymbol{x}_m)) - \Delta(\boldsymbol{A}_i, \boldsymbol{x}_n, \boldsymbol{H}_j(\boldsymbol{x}_n)) \right)$
   end For
3. $\boldsymbol{w} = (\boldsymbol{z})^+ / ||(\boldsymbol{z})^+||_2$
   where, $(\boldsymbol{z})^+ = [max(z_1, 0), ..., max(z_F, 0)]^T$

3.2.4 Iterative Search Margin-Based Algorithm with Side Constraints (Simba-Sc)

As mentioned in section 3.2.1, the Simba-Sc algorithm, proposed by Yang and Song [107], is obtained when using the l_2-norm with this mathematical interpretation. It is a modification of Simba algorithm (detailed in section 2.5.2.2 of chapter 2) into a semi-supervised margin-based algorithm that iteratively utilizes pairwise constraints specifically cannot-link ones to evaluate the ability of features in discriminating data points. It also uses a gradient ascent method to maximize its margin-based objective function. Thus, a higher score means a more relevant feature. Note that, Simba-Sc has a mechanism to deal with redundancy, however, it may still choose correlated features only when this contributes positively to the overall performance.

Definition 3.7. The constrained hypothesis-margin of a cannot-link constraint $(\boldsymbol{x}_n, \boldsymbol{x}_m)$ denoted by $\rho(\boldsymbol{x}_n, \boldsymbol{x}_m)$ is given by half the difference between the distance from the data point $\boldsymbol{x}_n$ to the nearhit of $\boldsymbol{x}_m$ and the distance to its own nearhit. However, similarly to Equation (2.23), we also omit the half obtaining the following:

$$\rho(\boldsymbol{x}_n, \boldsymbol{x}_m) = \|\boldsymbol{x}_n - \boldsymbol{H}(\boldsymbol{x}_m)\| - \|\boldsymbol{x}_n - \boldsymbol{H}(\boldsymbol{x}_n)\| \tag{3.17}$$

where the used distance function $\|\,.\,\|$ is the l_2-norm (or Euclidean distance) defined as in Equation (2.24).

Definition 3.8. By considering a training set $\boldsymbol{X}$, a cannot-link constraint $(\boldsymbol{x}_n, \boldsymbol{x}_m)$, and the weight vector $\boldsymbol{w}$, the weighted hypothesis-margin of $(\boldsymbol{x}_n, \boldsymbol{x}_m)$ is given by:

$$\rho((\boldsymbol{x}_n, \boldsymbol{x}_m), \boldsymbol{w}) = \|\boldsymbol{x}_n - \boldsymbol{H}(\boldsymbol{x}_n)\|_{\boldsymbol{w}} - \|\boldsymbol{x}_n - \boldsymbol{H}(\boldsymbol{x}_n)\|_{\boldsymbol{w}} \tag{3.18}$$

where $\|\boldsymbol{v}\|_{\boldsymbol{w}}$ is defined in Equation (2.26).

Definition 3.9. Considering the training set $\boldsymbol{X}$ and the feature weights denoted by the vector $\boldsymbol{w}$, the evaluation function of Simba-Sc is given by:

$$\rho(\boldsymbol{X}, \boldsymbol{w}) = \sum_{(\boldsymbol{x}_n, \boldsymbol{x}_m) \in CL} \rho((\boldsymbol{x}_n, \boldsymbol{x}_m), \boldsymbol{w}) \tag{3.19}$$

By analyzing Equation (3.19), the authors declared that a natural normalization constraint is required on $\boldsymbol{w}$, i.e., $max\ (w_i^2) = 1$ [107]. Besides, similarly to Simba, they ignore this normalization constraint during the calculation of $\boldsymbol{w}$ since for any scalar λ,

the weighed constrained margin $\rho((\boldsymbol{x}_n,\boldsymbol{x}_m),\lambda\boldsymbol{w}) = |\lambda|\rho((\boldsymbol{x}_n,\boldsymbol{x}_m),\boldsymbol{w})$. However, after finding $\boldsymbol{w}$, the normalization constraint $\|\boldsymbol{w}^2\|_\infty = 1$ should be projected on it.

Similarly to Simba, the building blocks of the evaluation function $\rho(\boldsymbol{X},\boldsymbol{w})$ of Simba-Sc are the weighted constrained margins of each cannot-link constraint known to us as $\rho((\boldsymbol{x}_n,\boldsymbol{x}_m),\boldsymbol{w})$. Thus, as $\rho(\boldsymbol{X},\boldsymbol{w})$ is smooth almost everywhere, the gradient ascent evaluated on the set of all cannot-link constraints is obtained similarly to Equation (2.28) as follows:

$$\begin{aligned}\frac{\partial\rho(\boldsymbol{X},\boldsymbol{w})}{\partial w_i} &= \sum_{(\boldsymbol{x}_n,\boldsymbol{x}_m)\in CL} \frac{\partial}{\partial w_i}\left(\|\boldsymbol{x}_n - \boldsymbol{H}(\boldsymbol{x}_m)\|_{\boldsymbol{w}} - \|\boldsymbol{x}_n - \boldsymbol{H}(\boldsymbol{x}_n)\|_{\boldsymbol{w}}\right) \\ &= \sum_{(\boldsymbol{x}_n,\boldsymbol{x}_m)\in CL} \left[\frac{(x_{ni}-H(x_m)_i)^2}{\|\boldsymbol{x}_n-\boldsymbol{H}(\boldsymbol{x}_m)\|_{\boldsymbol{w}}} - \frac{(x_{ni}-H(x_n)_i)^2}{\|\boldsymbol{x}_n-\boldsymbol{H}(\boldsymbol{x}_n)\|_{\boldsymbol{w}}}\right] w_i\end{aligned} \tag{3.20}$$

where $H(x_m)_i$ and $H(x_n)_i$ denote the values of the nearhits of $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$ on a specific feature $\boldsymbol{A}_i$ respectively.

Simba-Sc algorithm presented in **Algorithm 3.3** uses the following weight updating equation:

$$w_i^{new} = w_i^{old} + \left[\frac{(x_{ni}-H(x_m)_i)^2}{\|\boldsymbol{x}_n-\boldsymbol{H}(\boldsymbol{x}_m)\|_{\boldsymbol{w}}} - \frac{(x_{ni}-H(x_n)_i)^2}{\|\boldsymbol{x}_n-\boldsymbol{H}(\boldsymbol{x}_n)\|_{\boldsymbol{w}}}\right] w_i^{old} \tag{3.21}$$

Also similarly to Simba, the optimization method used in Simba-Sc is vulnerable to local maxima. Although it is restarted multiple times from different starting points, it still cannot guarantee obtaining a global maximum. Besides, as a constrained (mathematically) non-linear optimization problem, Simba-Sc also can not be easily solved through well-established optimization techniques. So, it first optimizes the margin-based objective function using a gradient ascent method while ignoring the normalization constraint. Then, as a separate final step, the obtained solution is projected over these constraints as can be seen in step (3) of **Algorithm 3.3**. In conclusion, the convergence of Simba-Sc algorithm is also doubtful [50].

Algorithm 3.3. Simba-Sc

Input:

- Training data $\boldsymbol{X}$
- Set of cannot-link constraints CL
- Number of iterations T

Output: Weight vector $\boldsymbol{w}$

1. Initialize $\boldsymbol{w} = (1,1,...,1)$
2. For $t = 1,...,T$
   (a) Pick randomly a cannot-link constraint $(\boldsymbol{x}_n, \boldsymbol{x}_m)$ from CL
   (b) Calculate $\boldsymbol{H}(\boldsymbol{x}_m)$ and $\boldsymbol{H}(\boldsymbol{x}_n)$ with respect to $\boldsymbol{X}\backslash\{\boldsymbol{x}_n\}$and the weight vector $\boldsymbol{w}$
   (c) For $i = 1,...,F$ calculate

   $$\Delta_i = \left(\frac{(\Delta(\boldsymbol{A}_i, \boldsymbol{x}_n, \boldsymbol{H}(\boldsymbol{x}_m)))^2}{\Delta_{\boldsymbol{w}}(\boldsymbol{x}_n, \boldsymbol{H}(\boldsymbol{x}_m))} - \frac{(\Delta(\boldsymbol{A}_i, \boldsymbol{x}_n, \boldsymbol{H}(\boldsymbol{x}_n)))^2}{\Delta_{\boldsymbol{w}}(\boldsymbol{x}_n, \boldsymbol{H}(\boldsymbol{x}_n))} \right) w_i$$

   End For
   (d) $\boldsymbol{w} = \boldsymbol{w} + \boldsymbol{\Delta}$

   End For
3. $\boldsymbol{w} \leftarrow \boldsymbol{w}^2 / \left\| \boldsymbol{w}^2 \right\|_\infty$
   where $\boldsymbol{w}^2 = ({w_1}^2,...,{w_i}^2,...,{w_F}^2)$ and $\left\| \boldsymbol{w}^2 \right\|_\infty = \max({w_1}^2,...,{w_i}^2,...,{w_F}^2)$

3.3 Feature Clustering in a Constrained Context for Minimizing Redundancy

Feature relevance is not the only aspect of a good feature selection method. In fact, the feature space can be composed of the following four groups of features [25]: (a) completely irrelevant, (b) redundant and weakly relevant, (c) non-redundant but weakly relevant, and (d) strongly relevant features. As the first two groups can significantly degrade the performance of learning algorithms and decrease their computational efficiency [29, 30], it is expected from a good feature selection method to be able to keep

features from within groups (c) and (d).

However, we noticed that a drawback in Relief-Sc, inherited from its basic supervised precursor Relief algorithm [66], is that it lacks the ability to deal with redundancy among features. Nevertheless, it is well known that eliminating redundant features is also an important aspect of feature selection.

Generally, in supervised and unsupervised contexts, multiple researches applied feature clustering with the aim of selecting a non-redundant and relevant feature subset. However, they tended to build their similarity matrices between features using information theoretic measures like mutual information [91], conditional mutual information [43] and maximal information coefficients [92]. It was also considered easy to transfer traditional data clustering methods to work for feature clustering, however, the challenge was in finding a suitable and meaningful definition of similarity notion between features [119].

Therefore, we propose to extend our semi-supervised feature selection method by a novel combination of feature clustering with hypothesis-margin maximization (Relief-Sc 3.2.2) to obtain a non-redundant feature subset, by which, its features are ranked in the order of their relevance to a target concept. For that, we benefit from the characteristics of non-parametrized sparse representation where it is possible to reconstruct each feature by the sparse linear combination of others [120]. This is done through solving a L_1-minimization problem. In fact, we may treat these sparse coefficients as similarity weights to build our features similarity matrix on top of which we apply clustering. For instance, we adopt an agglomerative single-link hierarchical clustering method. It clusters features progressively into larger groups forming a multi-level tree diagram called dendrogram. Cutting this dendrogram on a specific level, results in one of the possible clustering solutions by which the features within the same cluster (or group) play important roles in reconstructing each other and are thus assumed to be redundant in our scenario.

On the other side, multiple ways were used to select the most representative feature from each cluster like using lasso-type penalty factors [121], feature-class association measures [87] or simply selecting the cluster's centroid [92]. However, in our approach, we aim at representing each cluster by the feature that best maximizes a pairwise constraint-relevancy margin-based objective function. This maximization is quantized by assigning bigger weights to features that best contribute to enlarging a

semi-supervised distance metric called constrained hypothesis-margin. Besides, the overall approach aims at maximizing relevancy while minimizing redundancy, and to the best of our knowledge, no work has previously done that by combining feature clustering upon sparse representation with the constrained hypothesis-margin. To be precise, Simba-Sc [107] that is detailed in section 3.2.4 deals differently with redundancy, this will be illustrated in our experimental comparisons.

In the following sections, we detail each step of the proposed approach. For instance, in section 3.3.1, we explain how to represent the relationships between features to be used in clustering and in section 3.3.2, we explain the used hierarchical clustering method in our context. Finally, in section 3.3.3, we present the overall semi-supervised feature selection approach that combines both feature clustering and hypothesis margin maximization.

3.3.1 Feature Space Sparse Graph Construction

Sparse graph representation has received a great deal of attention in recent years [31, 122, 123], this is due to its ability to find the most compact representation of the original data and to preserve its underlying discriminative information [47]. In fact, the sparse representation model generally aims at representing a data point using as few as possible other data points within the same dataset (over-complete dictionary). Conventionally, some recent work utilized sparse theory to build the similarity matrix among data points by assuming that each point can be reconstructed by the sparse linear combination of other points [120, 124, 125]. On the contrary, in our work, the similarity graph adjacency structure and the corresponding graph weights are built simultaneously among features instead of data points [126, 127]. While computing the sparse linear coefficients by solving a L_1-norm regularized least squares loss problem, the most similar features as well as their estimated similarity weights to the reconstructed feature are identified. Hence, we obtain the feature-wise sparse similarity matrix that will be used in grouping features.
It is important to note that the main advantages of using the L_1-graph are the following:

- It can lead to a sparse representation which can enhance the efficiency and the robustness to noise in learning algorithms [47].

- While many clustering algorithms [32, 128] are very sensitive to some parameters when building their similarity graphs (like the performance of traditional

spectral clustering that is heavily related to the choice of the parameter sigma in Gaussian kernel), our graph construction is parameter free.

- It obtains both the graph adjacency structure and the corresponding similarity weights by L_1-optimization, while L_2-graphs usually separate them into two steps.

To mathematically formalize the problem, we consider the data matrix from its feature's perspective as presented in Equation (1.2). Therefore, to reconstruct each feature (attribute) $\boldsymbol{A}_i$ using as few entries of $\boldsymbol{X}$ as possible, we set $\boldsymbol{X}_i = \boldsymbol{X}/\boldsymbol{A}_i = [\boldsymbol{A}_1,...,\boldsymbol{A}_{i-1},\boldsymbol{A}_{i+1},...,\boldsymbol{A}_F]$ and then we solve an L_0-norm optimization problem as follows:

$$\min_{\boldsymbol{s}_i} \|\boldsymbol{s}_i\|_0 \quad s.t. \quad \boldsymbol{A}_i = \boldsymbol{X}_i\boldsymbol{s}_i \tag{3.22}$$

where $\|.\|_0$ denotes the L_0-norm, which is equal to the number of non-zero components in $\boldsymbol{s}_i$, $\boldsymbol{s}_i = [s_{i1},...,s_{i,i-1},0,s_{i,i+1},...,s_{iF}]^T$ is a F-dimensional coefficients vector in which the i^{th} element is equal to zero (implying that $\boldsymbol{A}_i$ is removed). The element s_{ij} $(i \neq j)$ denotes the contribution of any other feature $\boldsymbol{A}_j$ in reconstructing $\boldsymbol{A}_i$.

Note that, the solution of Equation (3.22) is NP-hard. Thus, a sparse vector $\boldsymbol{s}_i$ can be approximately estimated by the following L_1-minimization problem [125]:

$$\min_{\boldsymbol{s}_i \geq \boldsymbol{0}} \|\boldsymbol{s}_i\|_1 \quad s.t. \quad \boldsymbol{A}_i = \boldsymbol{X}_i\boldsymbol{s}_i, \quad \boldsymbol{1}^T\mathbf{s}_i = 1 \tag{3.23}$$

where $\|.\|_1$ denotes the L_1-norm and $\boldsymbol{1} \in \mathbb{R}^F$ is a vector of all ones values.

In fact, due to the presence of noise, the constraint $\boldsymbol{A}_i = \boldsymbol{X}_i\boldsymbol{s}_i$ in Equation (3.23) does not always hold. Thus, Liu and Zhang [31] mentioned a modified robust extension (invariant to translation and rotation) to mitigate this problem. It can be defined as follows:

$$\min_{\boldsymbol{s}_i \geq \boldsymbol{0}} \|\boldsymbol{s}_i\|_1 \quad s.t. \quad \|\boldsymbol{A}_i - \boldsymbol{X}_i\boldsymbol{s}_i\|_2 < \xi, \quad \boldsymbol{1}^T\mathbf{s}_i = 1 \tag{3.24}$$

where ξ represents a given error tolerance. The sparse vector $\boldsymbol{s}_i$ is computed for each feature $\boldsymbol{A}_i$. The optimal solution of Equation (3.24) for each data point $\boldsymbol{A}_i$ is a sparse vector $\hat{\mathbf{s}}_i$, this vector allows building the sparse re-constructive similarity matrix $\boldsymbol{S} = (\hat{s}_{i,j})_{F\times F}$, defined by:

$$\boldsymbol{S} = [\hat{\boldsymbol{s}}_1,...,\hat{\boldsymbol{s}}_i,...,\hat{\boldsymbol{s}}_F]^T \tag{3.25}$$

Note that, the obtained $\boldsymbol{S}$ is usually asymmetric, so we force symmetry using $\boldsymbol{S} = (\boldsymbol{S}^T + \boldsymbol{S})/2$.

The L_1-minimization problem of Equation (3.24) can be solved in polynomial time by standard linear programming methods [120] using publicly available packages such as SLEP (Sparse learning with efficient projections) package [129]. As the vector $\hat{\mathbf{s}}_i$ is sparse (a lot of its components have zero values and few have non-zero ones), the features in the dataset which are far from each other will have very small (zero or near zero) coefficients. This solution can reflect the intrinsic geometric properties of feature space. **Algorithm 3.4** summarizes the graph construction.

Algorithm 3.4. Sparse Graph Construction

Input: Training data $\boldsymbol{X}$
Output: Feature Similarity matrix $\boldsymbol{S}$
Initialize $\boldsymbol{S} = \mathbf{0}$

1. For each feature $\boldsymbol{A}_i$

   (a) Eliminate $\boldsymbol{A}_i$ from $\boldsymbol{X}$ as

   $$\boldsymbol{X}_i = \boldsymbol{X} / \boldsymbol{A}_i = [\boldsymbol{A}_1, ..., \boldsymbol{A}_{i-1}, \boldsymbol{A}_{i+1}, ..., \boldsymbol{A}_F]$$

   (b) Solve Equation (3.24) to obtain $\hat{\mathbf{s}}_i$

   (c) Iff the j-th entry of $\hat{\mathbf{s}}_i$ denoted $\hat{\mathbf{s}}_{i,j} \neq 0$

   set the weight $\boldsymbol{S}_{ij} = |\hat{\mathbf{s}}_{i,j}|$, $1 \leq i, j \leq F$

2. Force symmetry by $\boldsymbol{S} = (\boldsymbol{S}^T + \boldsymbol{S})/2$

3.3.2 Agglomerative Hierarchical Feature Clustering

As we mentioned before, a good feature selection algorithm is expected to find features that are most relevant in terms of discriminating data points between different classes while being least correlated to each other. The latter is similar to the general assumption of clustering where the data is partitioned such that points within the same cluster are as similar as possible to each other and as different as possible from points in other clusters [91]. Focusing on the idea of finding the least redundant (most diverse) features brought up clustering the features themselves instead of clustering data points [43, 89]. For instance, this minimized-redundancy among features can be

obtained by grouping them into different groups according to a similarity criterion then choosing one or multiple features to represent each group.
Among the four primary clustering categories that are hierarchical, density-based, statistical and centroid-based [130], we were interested in hierarchical clustering. It is useful when the structure of the dataset can hold nested clusters and does not require a predefined number of clusters as this algorithm outputs a tree diagram called dendrogram. The dendrogram that records the sequence of merges of clusters (features) into larger clusters, presents a multi-level grouping of these features [121]. Hence, depending on the cutoff level of the dendrogram, the number of obtained clusters can vary between 1 and F. Intuitively, at lower levels of the dendrogram we have the clusters of most redundant features that were first to be grouped. Thus, cutting on low levels, results in a higher number of clusters and therefore more cluster representatives i.e. a bigger output feature subset. However, cutting the dendrogram on higher levels results in a smaller number of clusters and thus fewer cluster representatives i.e. a smaller output feature subset. Hence, although choosing a high level of clustering ensures eliminating more redundancy, it could still cause more information loss. As a result, a good quality of clustering is closely related to the problem-adequate choice of the cutoff level. Therefore, we decided to cutoff when merging distances become large enough to create a second-level hierarchy, which means when clusters of features start being merged together instead of merging individual features. This is explained by our goal of reducing redundancy among features to a certain level without excessive compression that might lead to some information loss.

In summary, hierarchical clustering can have multiple methods for computing the distance between clusters (ward, complete, median, centroid, single and others), however, as we are working with features clustering and as we wish to merge the clusters on the most similar features (not their average nor on the farthest two features within a cluster i.e complete linkage), the agglomerative single-linkage hierarchical clustering was applied. It takes as input the $F \times F$ feature-wise similarity matrix denoted by $\boldsymbol{S}$ obtained from section 3.3.1. This algorithm initially assigns each feature to its own cluster, then finds the largest element s_{ij} remaining in $\boldsymbol{S}$, after that the corresponding two most similar clusters (or features) are merged based on s_{ij}. After each merging step, the similarity matrix is updated by replacing the two grouped clusters (or features) by the newly formed cluster in $\boldsymbol{S}$. This update can be expressed as $s_{h,ij} = \max_{\boldsymbol{s}}\{s_{hi}, s_{hj}\}$ Where $s_{h,ij}$ is the similarity between the cluster newly obtained by merging clusters (features) $\boldsymbol{A}_i$ and $\boldsymbol{A}_j$ and another cluster $\boldsymbol{C}_h$ (or feature $\boldsymbol{A}_h$). s_{hi} and s_{hj} are the respective similarities between cluster (or feature) pairs.

3.3.3 Proposed Feature Selection Approach

Our proposed approach that combines feature clustering with ReliefF-Sc, called (FCRSC), is a filter-type feature selection method as it does not depend on the performance of any learning algorithm while obtaining its ranked feature subset. In addition, one very important advantage of our method is that it is non-parametric, which means its performance is not vulnerable to being closely related to any tuned parameter. Moreover, we do not specify the number of clusters to be obtained from hierarchical clustering, but instead, we state a mechanism that choses the cutoff automatically such that no excessive compression nor trivial solutions are obtained.

To sum it up, first, we build the feature similarity graph on top of which we apply agglomerative hierarchical clustering. This graph is obtained through sparse coding, where the assigned similarity weights between features are in fact sparse coefficients indicating how much each feature contributes in reconstructing the other. It is very important to find a clustering solution C where features compression is not exaggerated (obtain very few clusters) nor underestimated (obtaining the trivial solution: each feature in its own cluster). Meanwhile, a weight is also assigned to each feature by ReliefF-Sc as it maximizes the semi-supervised margin-based objective function.

The significance of this approach lies in the last but most important algorithm called FCRSC. It starts with the two available ingredients, i.e., the clustering solution C obtained by hierarchical clustering and the weight vector $\boldsymbol{w}$ obtained by ReliefF-Sc. Then, for each of the clusters C_l in C, the number of features within C_l is evaluated. When a cluster has one and only one feature (considered not redundant at all), it is directly added to the chosen feature subset $\boldsymbol{F_s}$. However, when more than one feature is assigned to C_l, the features within C_l are sorted in the descending order of their corresponding margin weights given in $\boldsymbol{w}$. Thus, the feature with the highest weight (most relevant) is added to the feature set $\boldsymbol{F_s}$ and the rest are eliminated as they are judged to be redundant. After getting the representative feature from each cluster, these are sorted again in the descending order of $\boldsymbol{w}$ leading to the optimization of a two-fold objective problem, i.e, (1) minimizing redundancy between features in $\boldsymbol{F_s}$ and (2) maximizing relevancy between the features and the cannot-link constraints in CL. Thus, we obtain the ranked feature subset $ranked\boldsymbol{F_s}$. Note that, the number of features in $ranked\boldsymbol{F_s}$ will be equal to the number of obtained clusters denoted $card(C_l)$.

Algorithm 3.5. FCRSC

Input:

- Set of Clusters C obtained by Hierarchical clustering
- Weight vector $\boldsymbol{w}$ obtained by **Algorithm 3.2**

Output: Ranked Feature Subset $ranked\boldsymbol{F_s}$
Initialize: Feature set $\boldsymbol{F_s} = \phi$,

1. **For** each cluster C_l in C

   if $card(C_l) = 1$

   $\boldsymbol{F_s} \leftarrow \boldsymbol{F_s} \cup Current feature$

   end if

   if $card(C_l) > 1$

   Sorted $\leftarrow$ sort (features $\in C_l$, descending $\boldsymbol{w}$)

   $Current feature \leftarrow$ Sorted$(first\ row)$

   $\boldsymbol{F_s} \leftarrow \boldsymbol{F_s} \cup Current feature$

   end if

   end For

2. $ranked\boldsymbol{F_s} \leftarrow$ sort ($\boldsymbol{F_s}$, descending $\boldsymbol{w}$)

3. return $ranked\boldsymbol{F_s}$

3.4 Experimental Results

We present the experimental results of this chapter in two separate sections. The first section focuses on comparing Relief-Sc with the margin-based Simba-Sc algorithm in terms of classification accuracy and their relation with the set of provided constraints. On the other hand, the second section uses a different data experimental setup to compare ReliefF-Sc and FCRSC with other well-known filter algorithms in terms of the ability to select relevant features (expressed by classification accuracy) and the ability to choose a less redundant feature subset while maintaining accuracy.

3.4.1 Experimental Results on Relief-Sc: Selection of Relevant Features

In this section, we compare the classification accuracy obtained by the nearest neighbor classification algorithm after feature selection using our proposed Relief-Sc algorithm and other classical feature selection algorithms: Fisher score (supervised), Laplacian score (unsupervised), and Simba-Sc (semi-supervised).

3.4.1.1 Datasets Description

For fair evaluation, we use four well-known numeric UCI machine learning datasets [3] that were used in [40] and [107], including Sonar, Soybean, Wine and Heart. We also use another two high dimensional gene-expression datasets: ColonCancer [131] and Leukemia [132].
ColonCancer and Leukemia datasets aim at specifying the presence or absence of cancerous tumors, they are characterized by a very small number of data points compared to the number of features they possess. These two gene-expression datasets are an obvious example of a dataset suffering from the "curse of dimensionality".

Table 3.1 summarizes the main characteristics of each dataset. The second column of this table shows the number of data points, the third column shows the data dimension and the fourth column shows the number of classes. Also, the fifth column specifies the number of points in each class in order to allow observing whether the classes are balanced or not and the last column presents the number of constraints used by constrained algorithms. Before analyzing the results, we note that a dataset can have features lying within different ranges, this affects the performance of feature selection algorithms leading to unreliable outcomes. Thus, similarly to [30], we normalize the features of each dataset using min-max criterion to scale their values between zero and one. Besides, similarly to [40], we partition the datasets into two

Table 3.1: Datasets and their characteristics

Dataset	#points	#features	#classes	#points per class	#constraints
Sonar	208	60	2	97, 111	10
Soybean	47	35	4	10, 10, 10, 17	8
Wine	178	13	3	59, 71, 48	20
Heart	270	13	2	150, 120	20
ColonCancer	62	2000	2	40, 22	10
Leukemia	72	5147	2	47, 25	10

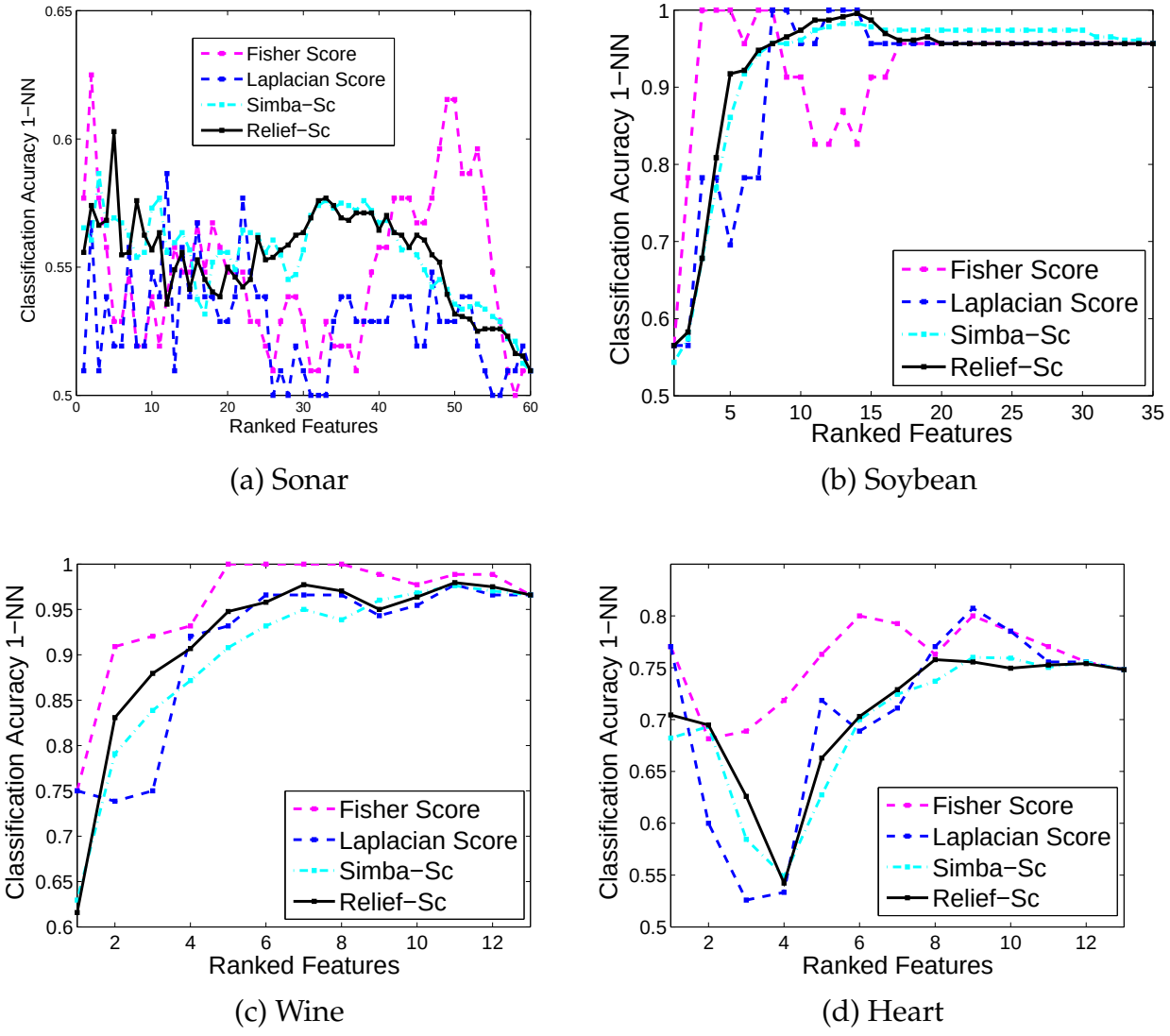
halves, the first half of data points from each class is considered for training, and the second half is considered for testing.
Feature selection is applied on the training subset, it allows ranking the features according to their assigned scores by different algorithms. Afterward, the classification accuracy of each ranked set of features is then measured by applying a classifier on the testing subset defined by these same features. As in [40, 107], we use 1-NN classifier with Euclidean distance for this purpose.

The comparison results between the two constrained algorithms Simba-Sc and our proposed Relief-Sc are averaged over 10 runs. This means that at each run Simba-Sc and Relief-Sc are applied on different cannot-link constraints (randomly generated). Since these constraints are considered more important than must-link ones from the view point of margin, the latter two algorithms utilized them. To be clear, in each run these constraints are generated as follows: we randomly choose a pair of data points from the training set, then we check to which class each point belongs, if it appears that these two points belong to different classes, they are considered the end points of a cannot-link constraint. This operation is repeated until we reach the desired number of constraints. This number was specified similarly to [107] for the datasets Sonar, Soybean, Wine and Heart as 10, 8, 20 and 20 respectively. We also use 10 constraints for both ColonCancer and Leukemia datasets as can be seen in the last column of Table 3.1.

3.4.1.2 Comparison of Classification Performances on UCI Machine Learning Datasets

Figure (3.2) shows the plots of classification accuracies vs. the desired number of features selected by our considered algorithms on the UCI datasets: Figure (3.2a) is for Sonar, Figure (3.2b) is for Soybean, Figure (3.2c) is for Wine and Figure (3.2d) is for Heart.

As a reminder, the unsupervised Laplacian Score uses data discriminative power and locality preserving ability to select features instead of any supervision information, unlike supervised Fisher Score that uses full class labels, yet both Laplacian and Fisher Scores still have to access the whole training data to calculate feature scores. In fact, semi-supervised Relief-Sc and Simba-Sc use supervision information a bit more than Laplacian Score, but much less than Fisher Score. This supervision information takes the form of few cannot-link constraints. The latter was reflected in Figure (3.2), where we can see that the classification accuracies of Relief-Sc and Simba-Sc generally lies between the classification accuracies of Fisher and Laplacian Scores respec-



(a) Sonar

(b) Soybean

(c) Wine

(d) Heart

Figure 3.2: Classification accuracy vs. different number of selected features on 4 UCI datasets: (a) Sonar; (b) Soybean; (c) Wine; (d) Heart. The number of constraints used for each dataset is presented in Table 3.1.

Table 3.2: The highest classification accuracy rates *Acc* (in %), with *d* representing the dimension of the considered selected feature space and *F* representing the original feature space.

	Simba-Sc		Relief-Sc		Without Selection	
	Acc	*d*	*Acc*	*d*	*Acc*	*F*
Sonar	58.65	3	**60.29**	5	50.96	60
Soybean	98.26	13	**99.57**	14	95.65	35
Wine	97.6	11	**97.9**	11	96.59	13
Heart	**76**	9	75.78	8	74.81	13

tively. Sometimes, the constrained algorithms can even reach close classification accuracies to that of supervised scores. Since the accuracy curves achieved by Relief-Sc and Simba-Sc are very comparable, we record their highest accuracy rates *Acc* and the corresponding dimension *d* of their selected feature subset in Table 3.2. We also show in this table the rates obtained by using the original feature space without performing feature selection.

From Table 3.2, we can first notice that the highest classification accuracy rates *Acc* achieved when applying the two constrained selection algorithms are greater than the rates achieved without selection (i.e. on the original feature space). This confirms the importance of feature selection in improving the classification results. Indeed, this is due to the fact that only few of the initially available features are usually highly relevant and not noisy with respect to the desired objective (e.g. classifying a person as healthy or diseased). From this table, we can also see that Relief-Sc was superior to its constrained competitor Simba-Sc almost always, this is true except on Heart dataset. Also, when Relief-Sc recorded higher accuracies, the selected feature space was very small compared to the size of the original space, especially for Sonar (5 out of 60) and Soybean (14 out of 35). Note that, the decrease in accuracy for Heart dataset (Figure (3.2d)) is caused by the presence of hybrid features. Qualitative features having nominal values contributed more to the margin calculation than the other quantitative features even if they are not really more relevant. A *ramp function* was suggested as a solution to this problem, by which, Δ is set to 0 (or 1) if the difference between values of continuous features is equal to or below (above) a user-predefined minimum (or maximum). In between these two boundaries, a function of the distance from them is used [37, 133]. However, in practice, it may be challenging to apply such an approach since it requires a problem-specific optimization of two additional user-defined parameters [2].

3.4.1.3 Comparison of Classification Performances on High Dimensional Gene-Expression Datasets

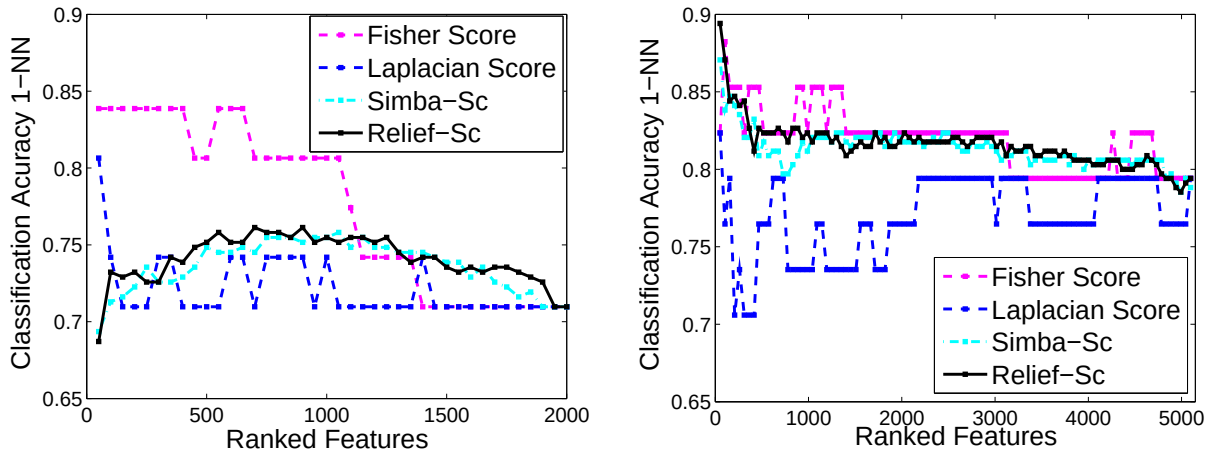
Figure (3.3) shows the plots of classification accuracies vs. the desired number of features selected by the considered algorithms on these datasets: Figure (3.3a) is for ColonCancer and Figure (3.3b) is for Leukemia. Also similarly to Table 3.2, we record the highest accuracy rates *Acc* and the corresponding dimension d selected by Relief-Sc and Simba-Sc on these two gene-expression datasets in Table 3.3.

For instance, Table 3.3 shows that Relief-Sc achieved a maximum accuracy of 76.45% on a set of 658 features out of 2000, unlike Simba-Sc that provided its highest accuracy of 76.13% on a set of 1008 features. This means the feature space can be reduced by more than half using Relief-Sc and still provide fair classification accuracy. On the other side, Figure (3.3b) shows the Leukemia dataset where the classification performance of Relief-Sc was not only comparable to the other constrained Simba-Sc algorithm but also to the supervised Fisher Score. The accuracy curve achieved by Fisher Score is close to that reached by Relief-Sc, i.e., 89.41% using only 52 features. Results

Table 3.3: The highest Classification Accuracy Rates *Acc* (in %), with *d* representing the dimension of the considered selected feature space and *F* representing the original feature space.

	Simba-Sc		Relief-Sc		Without Selection	
	Acc	*d*	*Acc*	*d*	*Acc*	*F*
ColonCancer	76.13	1008	**76.45**	658	70	2000
Leukemia	87.94	33	**89.41**	52	79	5147

on both UCI and gene-expression datasets show that integrating supervision information in the form of few cannot-link constraints only, can still be very useful for feature selection especially for the Relief-Sc algorithm. The classification performance of this algorithm generally lies between that of Fisher and Laplacian Scores respectively due to the amount of supervision information available for each algorithm, yet it proved that it can significantly reduce the size of the feature space while preserving good classification performance.



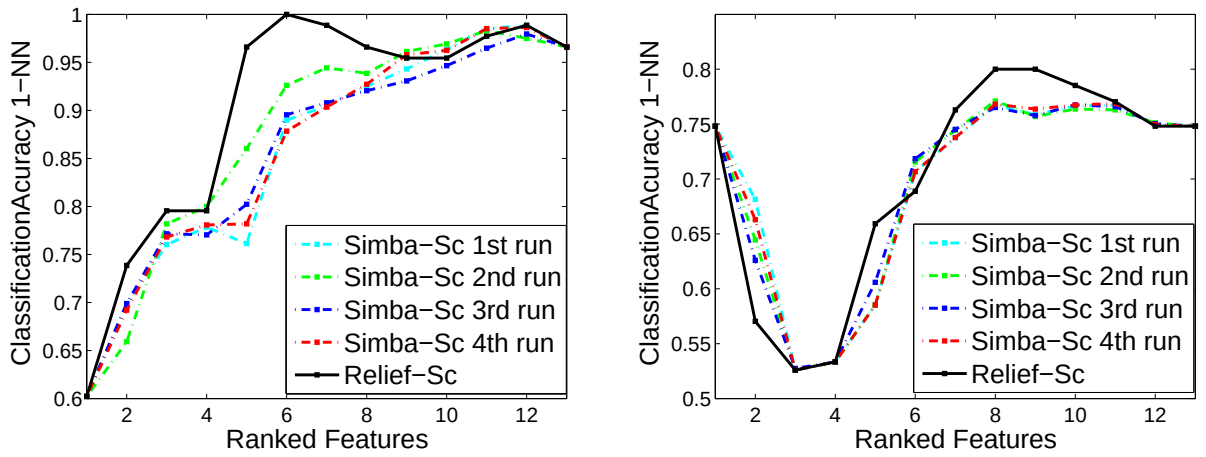
(a) ColonCancer (b) Leukemia

Figure 3.3: Classification Accuracy vs. different number of selected features on 2 high dimensional gene-expression datasets using 1NN: (a) ColonCancer; (b) Leukemia. The number of constraints used for each dataset is presented in Table 3.1.

3.4.1.4 Comparison of Obtained Solutions on a Fixed Set of Cannot-link Constraints

As explained in section 3.2.4, Simba-Sc uses a gradient ascent optimization method to find a weight vector $\boldsymbol{w}$ that maximizes the hypothesis margin. It is then important to note that in order to mitigate the problem of local optima, Simba-Sc chooses multiple starting points and runs the whole algorithm for each of them. The output of each loop (starting point) is the margin calculated over the features of our cannot-link constraints in CL. Thus, for each starting point another permutation of constraints is chosen by Simba-Sc and consequently we obtain a different margin. Finally, the margin having the largest sum is the chosen solution and its corresponding $\boldsymbol{w}$ is assigned as feature weights. Therefore, it is then clear that Simba-Sc is vulnerable to be trapped in local maximum. This is because no random solution can guarantee that the algorithm tried all possible constraint permutations.

Consequently, on a fixed constraint set, Relief-Sc has the advantage of achieving a unique solution compared to Simba-Sc. In other words, it always obtains the same solution whereas Simba-Sc fluctuates between multiple local maximum. In order to validate this advantage, Figure (3.4a) and Figure (3.4b) show the classification results on four different successive runs of Relief-Sc and Simba-Sc on Wine and Heart datasets while fixing the set of cannot-link constraints. The unique solution provided by Relief-



(a) Wine (b) Heart

Figure 3.4: Classification Accuracies after successive runs of Relief-Sc and Simba-Sc on (a) Wine; (b) Heart with a fixed set of cannot-link constraints.

Sc can be clearly stated compared to the alternating Simba-Sc solutions between better and worse. This confirms that Relief-Sc is the only constrained algorithm that exploits the potential of a highly nonlinear classifier (1-NN) to find the closed-form solution of a simple convex problem.

3.4.1.5 Comparison in terms of Running Time and Constraints number

In terms of computational complexity, the original supervised Relief [49] and Simba [102] are equivalent, their complexity is calculated as $O(TNF)$ where T is the number of iterations, N is the number of points and F is the number of features. However, when iterating over all training data points, i.e. when $T = N$, the complexity becomes $O(N^2F)$, this is why some works considered decreasing the size of used data points by smart selection of training instances [134]. Similarly, the constrained Relief-Sc and Simba-Sc have equivalent computational complexity but in our case $T = card(CL)$, where $card(CL)$ is the number of cannot-link constraints. When $card(CL)$ is high, one can also consider smart selection of constraints to prevent a high computational cost.

As we mentioned above, Simba-Sc chooses multiple starting points and re-evaluates the whole algorithm for each of them. It then compares their results, the one that obtains the margin of the largest sum, is the chosen solution. Although the number of starting points is a constant that is not considered in the big O notation of computational complexity, re-running and comparing multiple runs to find the best solution

Table 3.4: The average running time (in ms) for each of Relief-Sc and Simba-Sc vs. the number of cannot-ink constraints. Maximum number of constraints is 220 for ColonCancer and 297 Leukemia. Starting points fixed to default = 5 set by authors [107].

	ColonCancer		Leukemia	
#Constraints	Relief-Sc	Simba-Sc	Relief-Sc	Simba-Sc
5	27.4	103.7	129.8	329.6
15	33.3	252.9	134.7	590.9
25	39.3	393.4	152.8	880.7
35	45.4	551	168.6	1176.1
Max	149.3	3893.3	548.4	9297.2

affects the over all running time. It is clear that Simba-Sc is vulnerable to be trapped in local maximum. This is because no random solution can guarantee that the algorithm tried all possible constraint permutations. If this was the case, the computational complexity of Simba-Sc would increase and the algorithm might become intractable.

For validation, we measure the running time of our proposed Relief-Sc and of Simba-

Table 3.5: The average running time (in ms) for Relief-Sc and Simba-Sc versus the number of starting points. Maximum number of cannot-link constraints is 220 for ColonCancer and 297 Leukemia.

	ColonCancer		Leukemia	
#StartPoints	Relief-Sc	Simba-Sc	Relief-Sc	Simba-Sc
5	27.4	103.7	129.8	329.6
15	27.4	226.6	129.8	516.1
25	27.4	336.2	129.8	745.9
35	27.4	457.2	129.8	980.2
Max	27.4	2582.3	129.8	6852.1

Sc on the two high dimensional datasets (ColonCancer and Leukemia). The results are shown in Tables 3.4 and 3.5 where Relief-Sc appears scalable in terms of the number of constraints considered, whereas the running time of Simba-Sc increases linearly with the number of constraints and starting points. For instance, on ColonCancer, when the number of training data points available is 31, having 20 in class 1 and 11 in class 2, a total of maximum 220 cannot-link constraints can be obtained. If we wish to try all possible solutions and find the global maximum, Simba-Sc should also set the number of starting points to 220. During this process, the elapsed time and randomness make Simba-Sc intractable although our training data is small. Table 3.6 shows how the running time of Simba-Sc increases when tested on both ColonCancer and Leukemia with

their corresponding maximum constraints and starting points. Note that from Table 3.7 we can see that the running time of Relief-Sc is near to that of the unsupervised Laplacian Score and even when it uses all possible constraints it still takes less time than Fisher Score.

Table 3.6: The Running time (in ms) for Simba-Sc when considering the maximum number of cannot-link constraints and the maximum number of starting points i.e. 220 for ColonCancer and 297 for Leukemia. Each constraint is randomly selected without replacement.

#StartPoints = #Constraints	ColonCancer	Leukemia
220	1.7×10^5	-
297	-	5.6×10^5

Table 3.7: The Running times (in ms) for each of Fisher and Laplacian Scores averaged over 20 runs on the two high dimensional datasets.

	ColonCancer	Leukemia
Fisher	687.6	1708.8
Laplacian	43.4	110.9

3.4.2 Experimental Results on FCRSC: Selection of Relevant and Non-redundant Features

In this section, we compare the performance of our proposed FCRSC approach with some of the well-known state of the art feature selection methods. This comparison is applied in terms of classification accuracy, redundancy-removal ability, and execution time. The used datasets, feature selection methods, and classifiers are detailed in the following sections.
It is very important to note that, in this section, ReliefF-Sc is used, which means, the algorithm considers *K*-nearhits (sections 3.2.2 and 3.2.3 show the difference between Relief-Sc and ReliefF-Sc).

3.4.2.1 Datasets Description

To evaluate the proposed approach, in this section, six well-known benchmark datasets representing a variety of problems were used. These datasets include Wine and Sonar common with the previous experiments in section 3.4.1, and Breast Cancer Wisconsin

(Diagnostic) Data Set (WDBC), Ionosphere, Spambase, and Arrhythmia as additional datasets also from the UCI machine learning repository [3]. We summarize the main characteristics of each dataset in Table 3.8.
We use the same normalization min-max method mentioned in 3.4.1.1 to keep the features within the same scale. Also for Arrhythmia dataset where some of the feature values are missing, similarly to [30] we replace them with the average of all available values of the same corresponding feature. In addition, in the following experiments

Table 3.8: Datasets and their characteristics

Dataset	#points	#features	#classes	#points per class	#constraints
Wine	178	13	3	59, 71, 48	20
WDBC	569	30	2	357, 212	40
Ionosphere	351	34	2	225, 126	20
Spambase	4601	57	2	1813, 2788	100
Sonar	208	60	2	97, 111	20
Arrhythmia	452	279	13	245, 44, 15, 15, 13, 25, 3, 2, 9, 50, 4, 5, 22	20

we partition each dataset into 2/3 for training and 1/3 for testing. This process is repeated independently for 10 times and only the averaged results are recorded. In each run, feature selection followed by classifiers learning are applied on the training subset to allow ranking the features according to their assigned scores by different algorithms and then training the classifier on these same ranked feature sets. The classification accuracy that can be obtained by each ranked set of features (each feature selection method) is then measured by applying the learned classifier on the testing subset defined by these same features.

3.4.2.2 Used Feature Selection methods, Classifiers, and Parameter Setting

We use the Variance and Laplacian scores as they are widely-used well-known unsupervised filter methods for comparison [30, 40]. We also choose to compare with the supervised mRMR method since it aims at optimizing the same two-fold objective problem. ReliefF, Simba-Sc and ReliefF-Sc, on the other side, are all margin-based, similarly to the proposed FCRSC.
In addition, as the three semi-supervised constrained algorithms Simba-Sc, ReliefF-Sc and the proposed FCRSC are dependent on cannot-link constraints, these constraints are generated in each run (similarly to [107]) as explained in section 3.4.1.1. Regarding the number of constraints, we considered them relatively to the number of data points available in each dataset. Thus, it was set to 20 for Wine, Ionosphere, Sonar and Arrhythmia, 40 for WDBC, and 100 for Spambase dataset. Moreover, the number

of starting points for the nonlinear optimization method (gradient ascent) in Simba-Sc is set to its default value by the authors i.e 5. Also for fair comparisons, common parameters between different algorithms were set to the same values. For instance, Laplacian score, ReliefF, ReliefF-Sc and FCRSC had their neighborhood size set to 10 in all experiments (similarly to [30]) except for Spambase, which was set to 60 due to its large sample size. We also set the number of features in the subset obtained by mRMR to be equal to the number of clusters obtained by FCRSC.

We use more than one classifier of different decision-making natures and learning processes in order to provide a fair evaluation of used filter feature selection methods independently of the applied classification rules. These classifiers are: Nearest Neighbor classifier, SVM, Naive Bayes, and C4.5. Besides, these experiments can also be eye-opening on which classifier can be best used with the proposed feature selection method.

3.4.2.3 Performance Evaluation in terms of Classification Accuracy and Feature set Redundancy for Constrained Algorithms

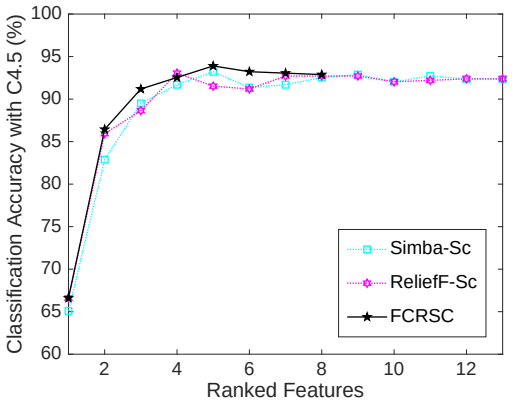
According to the previously detailed experimental setup, we first compare the performance of the proposed FCRSC method with that of the constrained feature selection methods stated in section 3.4.2.2. As these algorithms belong to the same supervision context of the proposed method and depend on an evaluation function of similar nature, we chose to closely compare with them at first. Therefore, the comparison is applied in terms of two important aspects: the classification accuracy a ranked feature set can bestow and the amount of redundancy that this set possesses. In fact, we aim at showing how FCRSC improves the performance of its precursor ReliefF-Sc and how it generally outperforms Simba-Sc in both of these aspects. In addition, C4.5 classifier was used as it can not detect feature interactions [135], a capability that the compared Relief-based algorithms has [97].

Hence, Figures (3.5) and (3.6) show the averaged accuracy rates obtained by the C4.5 classifier on the ranked feature sets obtained by the constrained filter methods (Simba-Sc, ReliefF-Sc and the proposed FCRSC) on Wine, WDBC, Ionosphere, Spambase, Sonar and Arrhythmia datasets over 10 independent runs. In fact, each two figures in a row correspond to one of the datasets. The figure on the left shows the averaged classification accuracy and the figure on the right shows the corresponding averaged representation entropy (i.e. an unsupervised metric used to compare redundancy in obtained feature subsets (section 1.7.1)).

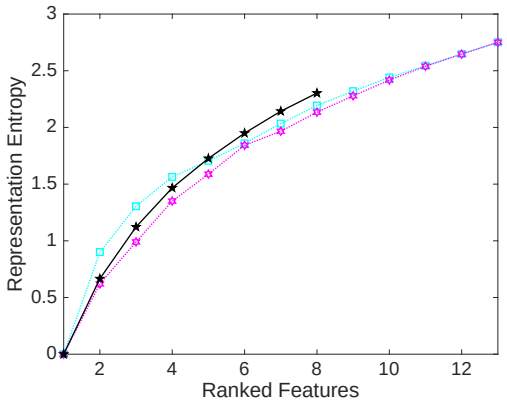
For instance, from Figures (3.5) and (3.6) we can see that generally FCRSC was always able to obtain a feature subset that provides a better classification accuracy while being less redundant. This was true except for Spambase in Figures (3.6a) and (3.6b), where the three algorithms performed approximately the same in terms of accuracy and redundancy and for Ionosphere in Figures (3.5e) and (3.5f), where FCRSC outperformed ReliefF-Sc and competed with Simba-Sc. On the other side, FCRSC proved that it can significantly improve the classification performance of its constrained antecedent ReliefF-Sc (e.g. Ionosphere, Sonar and Arrhythmia) through compromising between maximum relevancy and minimum redundancy in order to compose a subset that holds either weakly relevant but non-redundant features or strongly relevant ones.

Thus, from Figure (3.5a) on Wine dataset, we can see that FCRSC outperformed both constrained algorithms. Although, Figure (3.5b) showing Representation Entropy (RE) on Wine, shows that Simba-Sc had chosen less redundant features as the first three ones, still FCRSC outperformed it in terms of classification accuracy. In addition, the results on WDBC dataset were interesting, by which, Figures (3.5c) and (3.5d) show that FCRSC was better than ReliefF-Sc and Simba-Sc in both classification accuracy and redundancy reduction. For instance, FCRSC allowed a maximum classification accuracy of 94.60% after only 12 features out of 30 in the original space. Knowing that similarly to FCRSC, Simba-Sc also has a mechanism to remove redundancy, it is important to note that it keeps redundant features only when they still enlarge the margin. This means that Simba-Sc can still choose redundant features which explains why FCRSC generally obtains higher representation entropy among the chosen feature subsets (Figures (3.5) and (3.6)). Since the presence of redundant features can decrease learning performance, removing them allowed FCRSC to obtain higher accuracy rates as stated in Figure (3.5c) on WDBC. Moreover, FCRSC on Sonar dataset, as can be seen in Figures (3.6c) and (3.6d), outperformed Simba-Sc and ReliefF-Sc from the first few features until it reached its maximum of 75.94% on only 19 features out of 60 in the original space. However, later on, starting from the 27th chosen feature, Simba-Sc and ReliefF-Sc performed slightly better, although FCRSC maintained less redundant feature subsets throughout all of its ranked features.

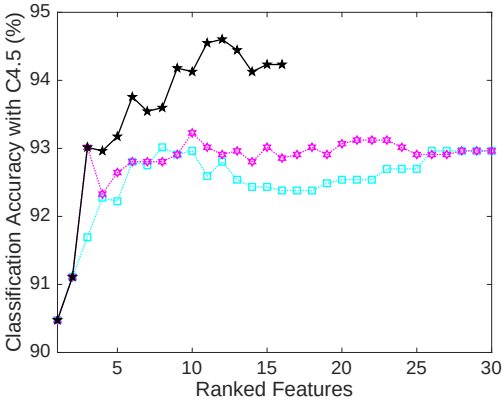
Finally, as can be seen in Figures (3.6e) and (3.6f) on Arrhythmia, FCRSC clearly outperformed Simba-Sc which was not able to detect a feature subset that can at least provide a classification accuracy equivalent to the one obtained without feature selection. In fact, ReliefF-Sc was able to find such a subset, however, its performance was lagging behind that of FCRSC. The latter was able to find a smaller feature subset with better classification accuracy and less redundancy among its features.



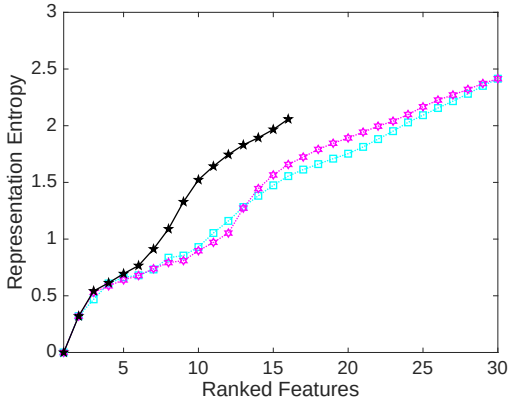
(a) Wine



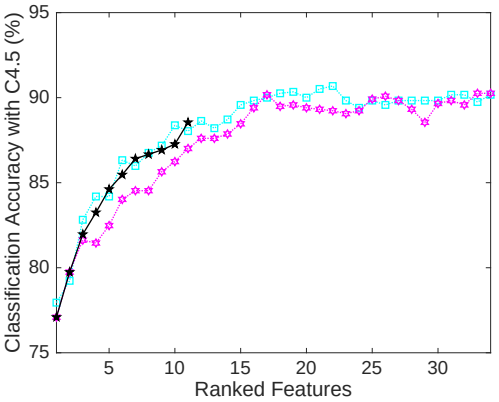
(b) Wine



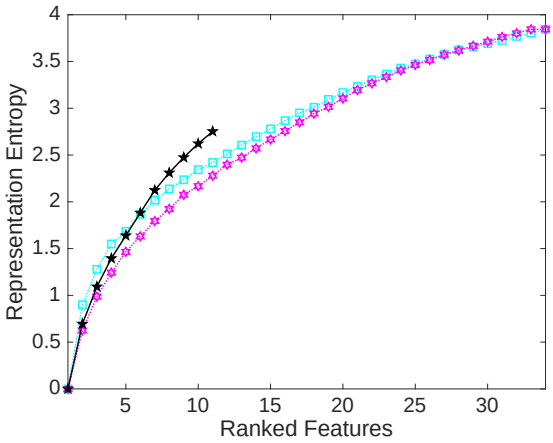
(c) WDBC



(d) WDBC

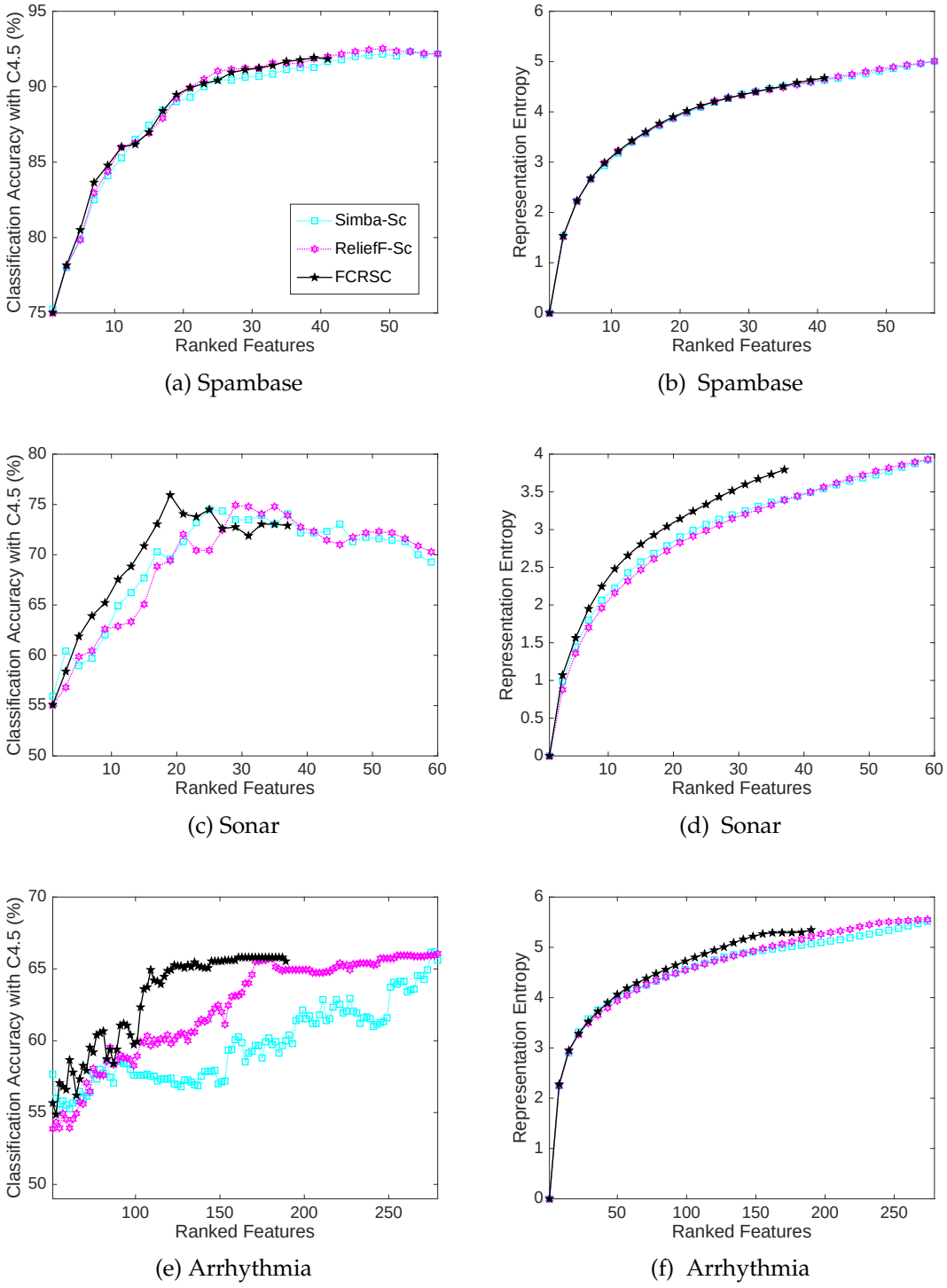


(e) Ionosphere



(f) Ionosphere

Figure 3.5: The averaged classification accuracy rates using C4.5 classifier vs. the number of ranked features obtained by the constrained algorithms: Simba-Sc, Relief-Sc and the proposed FCRSC over 10 independent runs on (a) Wine, (c) WDBC, and (e) Ionosphere datasets. The averaged Representation Entropy (RE) of each dataset is also shown in (b), (d), and (f) respectively.



(a) Spambase (b) Spambase

(c) Sonar (d) Sonar

(e) Arrhythmia (f) Arrhythmia

Figure 3.6: The averaged classification accuracy rates using C4.5 classifier vs. the number of ranked features obtained by the constrained algorithms: Simba-Sc, Relief-Sc and the proposed FCRSC over 10 independent runs on (a) Spambase, (c) Sonar, and (e) Arrhythmia datasets. The averaged Representation Entropy (RE) of each dataset is also shown in (b), (d), and (f) respectively.

3.4.2.4 Performance Evaluation in terms of Classification Accuracy using Multiple Classifiers for the Unsupervised, Supervised, and FCRSC Algorithms

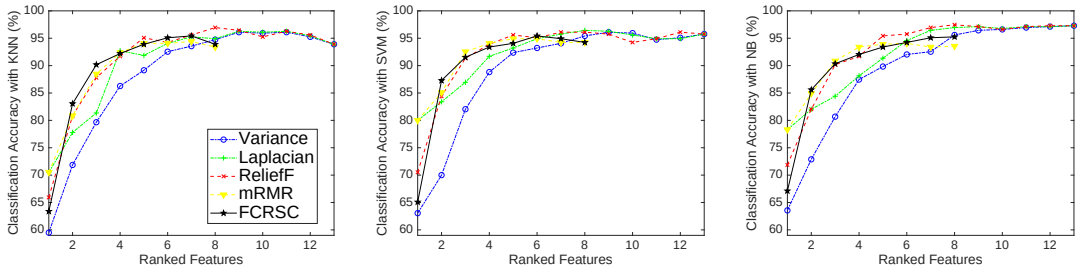
For the sake of generality, in this section, we compare the classification performance of the proposed constrained FCRSC method with that of the unsupervised and supervised methods mentioned in 3.4.2.2 using three different classifiers. We use the well-known *K*-NN, SVM and NB classifiers, each depending on a different decision rule nature, to show the general positioning of FCRSC performance with respect to some of the well-known feature selection state of the art algorithms. This also shows whether a performance degradation or improvement is classifier-dependent or is really imposed by the chosen feature subset.

For instance, each of the Figures (3.7) through (3.12) corresponds to a dataset with the classification accuracies obtained by three different classifiers upon the feature subsets ranked by Variance score, Laplacian score, ReliefF, mRMR and the proposed FCRSC.

From these figures, we can see that in general FCRSC was always able to obtain a higher accuracy curve compared to the unsupervised Variance and Laplacian scores except on WDBC where the five feature selection algorithms showed interfering and fluctuating accuracy curves from the first few ranked features as can be seen in Figure (3.8). Also, FCRSC was sometimes able to compete with supervised methods as can be seen in Figure (3.7) on Wine, Figure (3.11) on Sonar and Figure (3.12) on Arrhythmia.

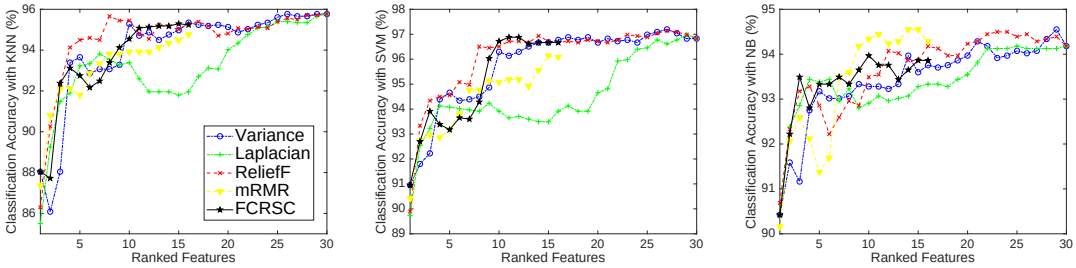
For instance, Figure (3.7) shows that using the three different classifiers on Wine dataset, FCRSC performed better than the unsupervised Variance score on all of them from the first few features. It also outperformed the Laplacian score, however, the latter chose a better starting feature. In addition, as mentioned before, FCRSC competed with the supervised ReliefF and mRMR as can be seen in Figures (3.7a) and (3.7b), where in fact, FCRSC and mRMR performed approximately the same. This can be due to their similar behavior in compromising between maximizing relevancy or minimizing redundancy.

Also, FCRSC on Ionosphere, as can be seen in Figure (3.9), clearly outperformed the unsupervised methods on the three classifiers in a very similar manner. Again, Figures (3.9a) and (3.9b) show a close classification accuracy values recorded by the supervised mRMR and the constrained FCRSC.



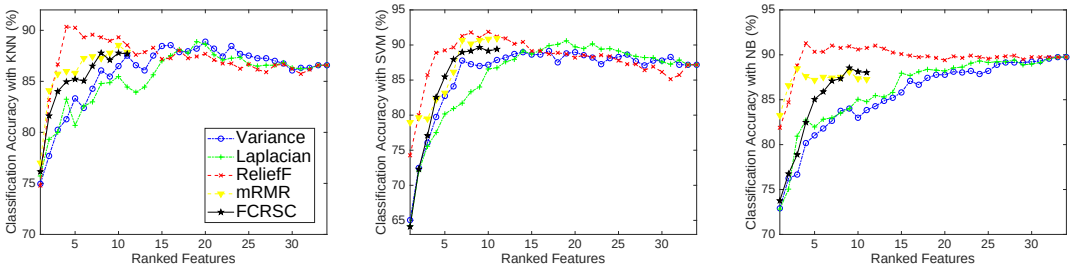
(a) (b) (c)

Figure 3.7: Wine dataset: the averaged classification accuracy rates using (a) *K*-NN, (b) SVM, and (c) NB classifiers vs. the number of ranked features obtained by Variance score, Laplacian score, ReliefF, mRMR and the proposed FCRSC over 10 independent runs.



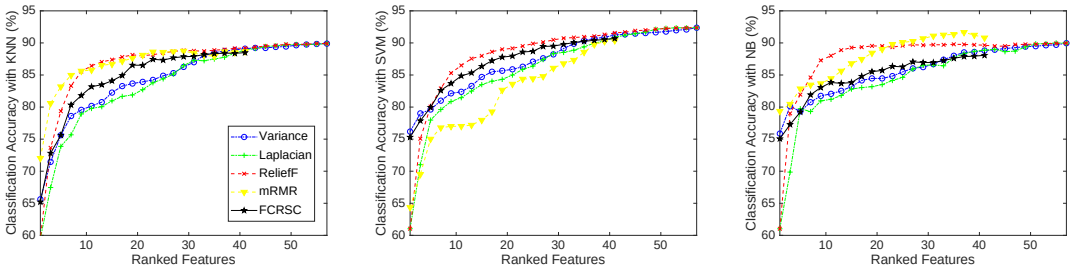
(a) (b) (c)

Figure 3.8: WDBC dataset: the averaged classification accuracy rates using (a) *K*-NN, (b) SVM, and (c) NB classifiers vs. the number of ranked features obtained by Variance score, Laplacian score, ReliefF, mRMR and the proposed FCRSC over 10 independent runs.



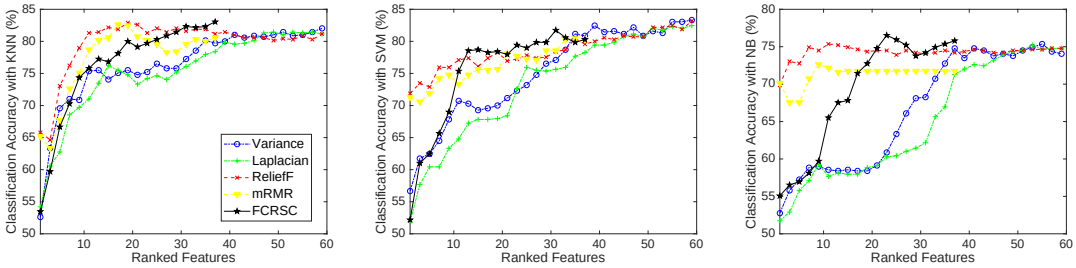
(a) (b) (c)

Figure 3.9: Ionosphere dataset: the averaged classification accuracy rates using (a) *K*-NN, (b) SVM, and (c) NB classifiers vs. the number of ranked features obtained by Variance score, Laplacian score, ReliefF, mRMR and the proposed FCRSC over 10 independent runs.



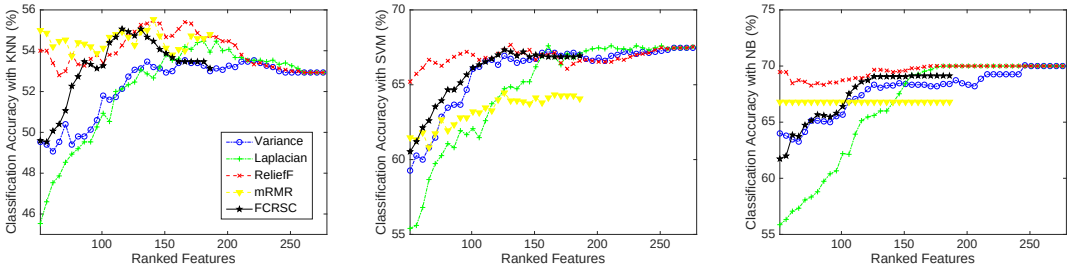
(a) (b) (c)

Figure 3.10: Spambase dataset: the averaged classification accuracy rates using (a) *K*-NN, (b) SVM, and (c) NB classifiers vs. the number of ranked features obtained by Variance score, Laplacian score, ReliefF, mRMR and the proposed FCRSC over 10 independent runs.



(a) (b) (c)

Figure 3.11: Sonar dataset: the averaged classification accuracy rates using (a) *K*-NN, (b) SVM, and (c) NB classifiers vs. the number of ranked features obtained by Variance score, Laplacian score, ReliefF, mRMR and the proposed FCRSC over 10 independent runs.



(a) (b) (c)

Figure 3.12: Arrhythmia dataset: the averaged classification accuracy rates using (a) *K*-NN, (b) SVM, and (c) NB classifiers vs. the number of ranked features obtained by Variance score, Laplacian score, ReliefF, mRMR and the proposed FCRSC over 10 independent runs.

On the other side, on Spambase dataset, as can be seen in Figure (3.10), feature selection was generally not significant by all algorithms (the best accuracy was obtained on the original feature set) except for ReliefF and mRMR with NB classifier shown in Figure (3.10c). However, the performance of FCRSC was generally stable with the three used classifiers. It lies between the supervised and unsupervised methods except for SVM classifier presented in Figure (3.10b) where both the unsupervised methods together with FCRSC outperformed mRMR. In addition, the results on Sonar and Arrhythmia were interesting, where as can be seen in Figure (3.11a) on Sonar dataset, both Variance and Laplacian scores reached their highest accuracy rates (82.03%) on the full feature space of Sonar dataset i.e 60 features, whereas FCRSC obtained an accuracy of 83.04% over only 37 features. This shows that, in this case, Variance and Laplacian scores could not obtain a smaller feature subset that can provide a similar or better classification than the original one. It is important here to note that, the performance degradation of FCRSC that appeared between approximately the 27th and 37th features on Sonar with C4.5 classifier (analyzed in the previous section and presented in Figure (3.6c)) was classifier-related since a good performance was obtained for the same features using *K*-NN, SVM and NB classifiers.

On the other hand, Figure (3.12) on Arrhythmia, also shows that FCRSC outperforms the unsupervised algorithms and competes with the supervised ones. In fact, it was also able to obtain either a similar or a higher classification accuracy compared to the one obtained by the full feature space with approximately only half the number of features. For example, an accuracy of 55.3% on 117 feature was recorded using FCRSC with *K*NN (Figure (3.12a)) compared to 52.93% on 279 features without feature selection and 55.5% on 141 features using ReliefF and mRMR.

In conclusion, FCRSC aims at finding a relevant and non-redundant feature subset that is said to either maintain the classification accuracy obtained on the full feature space **or** provide an enhanced accuracy performance (through removing the irrelevant and noisy features). It is important to note that although FCRSC chooses a final subset of features (size($\boldsymbol{F_s}$) $< F$), it is still a ranking feature selection method, which means, similarly to the other feature selection methods mentioned in this paper, subsets that are also smaller than $\boldsymbol{F_s}$ can be obtained. Moreover, as FCRSC utilizes the ReliefF-Sc algorithm for its margin maximization objective, it was clear from the results that removing redundant features in addition to irrelevant ones allows better classification performance. Finally, using more than one classifier with different decision-making natures provided a fair evaluation of the used filter feature selection methods and

proved the general independence between them and the classifiers.

3.4.2.5 Comparison in terms of Running Time

In this section, we consider the different supervised, unsupervised and semi-supervised feature selection methods from the execution time perspective. Thus, Table 3.9 presents the average execution time (in ms) of each feature selection method over 10 independent runs.

Table 3.9: Average execution time (in ms) of the different Unsupervised, Supervised, and Semi-supervised feature selection methods over 10 independent runs.

Dataset	Feature Selection method						
	Unsupervised		Supervised		Semi-supervised		
	Variance	Laplacian	ReliefF	mRMR	Simba-Sc	ReliefF-Sc	FCRSC
Wine	0.41	9.06	71.62	13.66	569.04	1.70	9.61
WDBC	0.81	73.68	205.70	43.25	3166.88	5.97	47.31
Ionosphere	0.34	29.66	128.20	28.39	1032.83	2.50	41.80
Spambase	1.92	5408.62	15290.15	342.65	67081.99	755.62	1905.42
Sonar	0.59	16.06	108.83	35.76	694.99	2.73	89.55
Arrythmia	2.46	93.64	4727.31	446.71	1336.32	5.47	1463.52
Average	1.09	938.45	3421.97	151.73	12313.67	129.00	592.87

For instance, from this table we can see that, Variance score, as expected, had the least execution time with the least differences between datasets since it is the simplest method among all. However, Laplacian, ReliefF and Simba-Sc appeared to have a great increase in execution time as the number of data points increase significantly as was the case between Spambase (4601 data points) and Sonar (208 data points) with approximately equal number of features. Although ReliefF-Sc and FCRSC also had a significant increase in the latter case, this increase was less steep.

Noting that Simba-Sc, ReliefF-Sc and FCRSC are all dependent on constraints and are provided with the same number on each dataset, we declare that Simba-Sc consumes much more time due to it being repeated from 5 different starting points (gradient ascent method) to optimize its objective function. On the other side, as the number of features increase significantly between 13 for Wine and 60 for Sonar (having close number of data points and same number of provided constraints i.e 20) the execution time by all the algorithms increased reasonably, however, ReliefF-Sc increased very little compared to FCRSC, and this is due to the time needed by FCRSC to apply the

steps of feature clustering and cluster-representative choosing.

Hence, the overall average execution time for each method on all datasets, as can be seen in the last row of Table 3.9, shows that FCRSC was generally faster than Laplacian, ReliefF, and Simba-Sc. Thus, in terms of computational complexity, we can say that FCRSC is mainly related to the following:

- the number of cannot-link constraints $card(CL)$
- the number of data points N
- the dimension of feature space F

In big O notation, the constrained ReliefF-Sc can be calculated in $O(card(CL) \times NF)$. Also, the ranking step that is used within all ranking feature selection methods needs $O(F \times \log(F))$. The proposed FCRSC is divided into multiple steps some of which can be done in parallel (clustering the features and calculating their margin weights). For instance, the construction of the sparse graph costs $O(F^2)$ [47], the single linkage hierarchical clustering of features also needs $O(F^2)$. Hence, the overall computational complexity by FCRSC can be calculated as $O(Max(card(CL) \times NF, F^2, F \times log(F)))$.

3.5 Conclusion

To deal with data high dimensionality in a semi-supervised context, we discussed margin-based feature selection algorithms with side pairwise constraints. Considering only cannot-link constraints due to their conceptual importance from the margin's perspective and their ease of acquirement, we suggested Relief-Sc algorithm. It utilizes the cannot-link constraints to assign weights to different features according to their contribution in maximizing the hypothesis-margin. Relief-Sc is capable of finding a unique relevant feature subset in a closed-form. Experimental results on multiple UCI machine learning and two high dimensional gene-expression datasets proved that Relief-Sc can obtain comparable performances with supervised Fisher Score using full class labels on the entire training set, and has better performance than the other constrained Score Simba-Sc. Note that in [107], they concluded that Simba-Sc had better performance than Constraint Score-1 and Constraint Score-2, this means Relief-Sc also has better performance than these two scores. Moreover, Relief-Sc can work only on few constraints from the training set, which ensures computational advantage over other algorithms when dealing with high dimensional data. A robust version of Relief-Sc, called ReliefF-Sc, was also suggested.

In addition, a novel combination of feature clustering upon a sparse graph with a margin-based objective function called FCRSC was proposed. This approach is said to handle the two core aspects of feature selection (relevancy and redundancy) in three main building blocks where it first constructs the similarity matrix between features through sparse representation. Secondly, on top of the latter, feature clustering is applied simultaneously with the application of the margin-based algorithm ReliefF-Sc. Finally, FCRSC obtains its final feature subset by choosing the feature that most enlarges the margin from each cluster of features, hence, maximizing relevancy while minimizing redundancy. The efficiency of this approach was compared to supervised, unsupervised and semi-supervised filter feature selection methods using four different classification schemes on six UCI well-known benchmark machine learning datasets. The results showed the satisfactory performance of FCRSC that outperformed the unsupervised and semi-supervised methods on most datasets and also competed with the supervised ones.

Chapter 4

Active Learning of Pairwise Constraints

Contents

4.1 Introduction

As explained in the previous chapter, our proposed algorithms Relief-Sc, ReliefF-Sc and FCRSC are all constrained semi-supervised algorithms that particularly utilize

cannot-link constraints. As a reminder, pairwise constraints are a cheaper kind of supervision information that does not need to reveal the class labels of data points. Initially, constraints were suggested to augment the performance of traditional clustering algorithms by increasing the semantic meaningfulness of the resulting clusters [136]. Accordingly, many algorithms used them to enhance clustering tasks [137–141], but only few considered using them for feature selection (e.g. [107, 142]).

However, in most current methods, pairwise constraints are said to be provided passively and generated randomly over multiple algorithmic runs by which the results are averaged. This leads to the need for a large number of constraints that might be redundant, unnecessary, and under some circumstances even inimical to the algorithm's performance. It also masks the individual effect of each constraint set and introduces a human labor-cost burden. Although it was assumed by the machine learning community that adding constraints as supervision information will guide and improve clustering performance, Davidson *et al.* [57] stated that the latter can still be degraded even when constraints are random but directly generated from data labels. It is then expected that the used constraints can affect the performance of constrained feature selection methods just like they affect constrained clustering.

Therefore, in this chapter, we suggest a framework for actively selecting and then propagating these constraints for feature selection. This framework has three main components. (1) The feature selection algorithm that utilizes constraints (detailed in sections 3.2.2 and 3.2.3, (2) our core contribution i.e. the process of selecting these constraints, and (3) the augmentation of supervision information through propagating them. In this chapter, we introduce the second two components:

- First, we target the process of systematically and actively selecting the constraints based on the idea of matrix perturbation [143]. Accordingly, we benefit from the spectral characteristics of the graph Laplacian which by turn is defined on the similarity matrix. The impact of each data couple (pairwise constraint) on this matrix can be reflected by the change it can cause to the graph Laplacian, especially to its eigensystem. When a small perturbation in the similarity value of a couple is able to perturb the graph Laplacian leading to a more well-separated form of the second eigenvector, this couple is definitely considered more important and significant as a constraint. Therefore, we propose an Active Constraint Selection (ACS) method based on a second-eigenvector sensitivity criterion. It aims at seeking those data couples with the highest ability to change the position of the most uncertain points (i.e. near zero) on the second eigenvector to

one of the two certain groups, thus, dividing the data into two well-separated groups. The latter couples will be the only ones queried for constraints, hence, reducing the cost of queries. In fact, these constraints that can decrease uncertainty and enhance class-separation, allow the selection of relevant features with the highest class-discriminative ability.

- Second, as we aim to query an oracle for the constraints, we are able to ask only a specific number of questions before this process gets inefficient and not user-friendly. Furthermore, for the family of margin-based algorithms, a sufficient set of points is needed to calculate the margin [2]. Thus, we adapt an exhaustive propagation method that allows spreading the constraints to the neighborhood of our initially instance-level selected ones. As this propagation produces soft constraints between data points, we also suggest thresholding for pruning the results and avoiding noisy constraints.

These two components together with ReliefF-Sc (section 3.2.3) form our proposed framework of active constraint selection and propagation for feature selection.

In order to validate our framework, we apply a series of experiments. First, we compare the classification accuracies of our proposed ReliefF-Sc with random constraints generation (RCG) to itself but with active constraint selection (ACS). Then, the latter is compared to Relief-Sc with the propagation of active constraint selection (PACS). Second, we compare the accuracies obtained by different well known supervised, unsupervised, and constrained feature selection algorithms when utilizing RCG and ACS. These experiments are applied on multiple well-known UCI machine learning datasets and two high-dimensional gene expression ones.

This chapter is structured as follows: in section 4.2, we explain some related work to constraints selection and propagation. In section 4.3, we explain our active constraint selection method while providing the needed background. In section 4.4, we detail our adaption of the constraint propagation method. Section 4.5 discusses the computational complexity of our method. The experimental results on the performance of feature selection after constraint selection and propagation are discussed in section 4.6. Finally, we conclude in section 4.7.

4.2 Related Work

Since it is much easier to obtain pairwise relations between data points than to obtain their class labels, many pieces of research considered solving real-life problems using them instead. As a result, it was important to navigate through available methods to understand how a more beneficial constraint set can be chosen.

4.2.1 Constraints Selection

Davidson *et al.* [57] suggested two measures namely "informativeness" and "coherence" of constraints in order to measure their utility, they also stated that the degraded performance after using a constraint set is not due to noisy or erroneous constraints but instead due to the interaction between this set and specifically the characteristics of the used algorithm. Although the two latter suggested notions were tailored to a clustering framework, they could be applied as two standalone measures independently of the learning model. Therefore, Benabdeslem and Hindawi [54] adopted the notion of "coherence" to measure the utility of constraints for their feature selection constrained Laplacian Score. Note that "informativeness" and "coherence" ensured "Direct constraint satisfaction" by which a must-link constraint is treated in a way that its two endpoints are close to each other and a cannot-link constraint is treated in a way that its two endpoints are ensured to be well-separated.

Furthermore, instance-level constraints can be generated from background knowledge given by an expert i.e querying an oracle, or by acquiring them from a set of labeled data [138, 144]. The second way of generating constraints was more actually used except when actively choosing constraints is the goal [139]. In fact, whatever way we choose to acquire constraints, measuring their utility is a good practice. Abin and Beigy [140] measured constraints utility actively using two assumptions. The first one states that a constraint is considered more informative if it is a cannot-link constraint joining two close data points, whereas the second assumption states that a constraint is considered more informative if it is a must-link constraint joining two far data points. The latter two assumptions give important spatial information about the data. Moreover, they used a Support vector data description (SVDD) to find the support vectors in the dataset and then generate constraints only from the points that are on the boundary, thus shaping the data. To make the best of the chosen constraints, they avoided information redundancy by choosing distant constraints. This means that when two constraints of the same type are available in the same vicinity, only one of them is considered.

Similarly, [145] suggested active spectral clustering method "Access" that examines the eigenvectors of the graph Laplacian. It uses the theoretical properties of spectral decomposition to identify data items that are located on the boundaries of clusters (boundary points and sparse points). It aimed to reduce the human interaction burden by actively obtaining constraints on these particular boundary points. However, it was closely related to the assumption of having data with very close clusters or overlapping boundaries and that the only reason for degraded performance is boundary points.

Another significant approach proposed by Wauthier *et al.* was to actively guide a spectral clustering algorithm to incrementally measure similarities that are most likely to reduce uncertainty [141]. Particularly, they study the trade-off between desired clustering quality and the amount of available information about the data. This trade-off is studied in the framework of spectral clustering by which not all pairwise similarity measurements are available initially. For that, they maintain estimations of true similarities as baselines. Then, they iteratively query an external black-box (e.g. a human expert) on points that show the highest uncertainty in the clustering task. It was stated that most realistic pairwise similarity matrices do exhibit some clustering structure even when incomplete. However, the selection of the most uncertain data pair, in this case, can be seen as a constraints selection problem by showing that the utility of having information of this particular couple is high.

4.2.2 Constraints Propagation

Generally, as the number of constraints is quadratically related to that of data points, it is only feasible to query for a small proportion of all possible constraints before the process turns to be resource-expensive, even when we utilize an inexpensive oracle [146]. Thus, in real-life scenarios, we might not be able to ask for a large number of queries and might have to work with the minimum but sufficient amount of data from the "constraint selection" step.

In [147], kamvar *et al.* just trivially modified the values of the similarity matrix to 1 and 0 where a must-link or a cannot-link constraint is present respectively. This method barely affects spectral clustering results because only similarities between constrained pairs are changed. In order to allow pairwise constraints to exert a stronger effect on spectral clustering, Lu and Ip [148] presented an exhaustive and efficient method for constraint propagation. They applied propagation by decomposing the

process into sub-propagation-problems and then solving each of the subproblems independently based on local K-NN graphs. It was noted that this method was the first to tackle constraint propagation to the entire dataset with computational efficiency. Moreover, as propagation is a pre-step done independently from the clustering method, it can be used with different constrained clustering methods only by adjusting the affinity matrix accordingly. This method is not limited to two-class problems, allows soft constraints, and results in a closed-form solution.

From a different point of view, Klein *et al.* [144] tackled space-level information given by constraints instead of only satisfying individual instance-level constraints, which means, a cannot-link constraint not only tells that its endpoints should be in different clusters, but also their nearby points should be similarly separated. They also propagated must-link constraints by modifying the actual distance or similarity graph and then applying unsupervised clustering on the modified graph. This implicitly holds the neighborhood-based constraints information. However, they stated that it was harder to maintain cannot-link constraints and used a proximity-based clustering algorithm to propagate their information.

On the other hand, Lu and Carreira-Perpinan [137] also researched constraint propagation and suggested a method for affinity propagation that utilizes the limited available pairwise constraints to change their corresponding values within the affinity matrix according to their positions through a Gaussian process. This method deals with multi-class problems through applying an heuristic search over the different cannot-link constraints. Another idea was to extend an exemplar-based clustering algorithm [149] to work in a semi-supervised context with propagating instance-level constraints [150]. This method modified the factor-graph to include meta-points (MTPs) that can take the role of exemplars. For instance, by enforcing constraints on these MTPs and then changing the similarities between points accordingly, affinity propagation is obtained. Furthermore, a semi-definite programming problem (SDP) that is also not limited to two-class problems was used by [151] to apply such constraint propagation but incurring a high computational cost.

When working in a semi-supervised context *consistency* is an important assumption. It means that points in the same neighborhood or on the same manifold are considered belonging to the same cluster. The latter, also known as cluster assumption, takes two different levels. When a neighborhood-based algorithm is used, the level is said to be local, whereas when the algorithm is structure-based, the level is

said to be global. For instance, Zoidi *et al.* [152] modified the method suggested in [153] to handle propagating pairwise constraints within a "Multiple Locality Preserving Projections with Cluster-based Label Propagation" (MLPP-CLP) framework, a new method for propagating person identity labels on facial images, this method used two types of knowledge: a projection matrix that holds the locality information from multiple image representations and a priori pairwise must-links and cannot-links. During the projection phase, both the propagated constraints and the Locality Preserving Projections (LPP) are used to find a projection matrix that respects pairwise constraints, maps data to a space of reduced dimensionality and retains the locality information of data. For instance, data that are must-linked should be mapped close to each other and data that are cannot-linked must be mapped far away from each other. To propagate constraints, they modified the smooth iterative graph-based label propagation function that utilizes an initial set of given labeled points to predict the labels of unlabeled ones. The input of the function was changed to be the initial constraint matrix instead of the initial label matrix, thus, the output is a neighborhood of propagated constraints instead of a matrix of predicted labels. It finally converges to a closed-form solution of fully propagated constraints. Nevertheless, this method finds application when there are multiple representations of data such as the case of multi-view camera systems.

From a different perspective, Wang *et al.* [154] answered the question of how to learn constraints through self-learning, thus transforming the work from expert teaching to self-teaching. The latter transition tackles the problem of incrementally polling an oracle for new constraints and minimizes the number of queries needed from the human experts. Briefly, they suggested extending the objective function of spectral clustering such that new constraints can be self-taught according to the graph affinity matrix and the few available prior knowledge (constraints). This work is considered as "transferring knowledge" since unconstrained data is gaining information through augmented constraints. But unlike other transfer learning settings, this framework has one domain only (affinity graph) and one task (clustering). It was claimed that this framework was the first to combine graph min-cut with matrix completion into an iterative self-teaching process. As they explained, spectral clustering had many advantages over other clustering schemes like its simple objective function, its easy implemented closed-form solution, and its ability of modeling arbitrarily shaped clusters. Yet, it greatly depends on the completeness and correctness of the data graph.

4.3 Active learning of Constraints

Since feature selection by ReliefF-Sc closely depends on the used constraint set, we extend to study thoroughly the systematic and active selection of the constraints that will be used by this algorithm. Accordingly, we focus on cannot-link constraints as they are considered more important than must-link constraints from the margin's perspective.

As we mentioned before, pairwise constraints were mostly used in constrained clustering problems to actively improve and guide clustering [139, 144, 145, 155]. Only few used them similarly in feature selection [107, 142]. As mentioned before, utilizing pairwise constraints does not necessarily improve clustering performance in the sense of increased agreement between the estimated clusters and the actual ones [57]. Even when some constraint sets are generated from data labels directly, they can still decrease clustering performance. Moreover, Davidson *et al.* [57] proved that the performance degradation or stability is not always due to false constraints, but sometimes due to ill interactions between a chosen constraint set and the objective function of the clustering algorithm used. Later, many pieces of research aimed at understanding how a more beneficial constraint set can be chosen [58, 140, 141].

Therefore, in the following sections, we first give a brief background on how we use graph Laplacian in 4.3.1 and then we detail the process of selecting constraints based on the graph Laplacian eigen-decomposition for feature selection instead of clustering in 4.3.2.

4.3.1 Using Graph Laplacian

This section closely depends on the definitions in section 1.3.2. For instance, for the dataset $\boldsymbol{X}$ with N points, we have an NxN symmetric similarity matrix $\boldsymbol{S}$ that holds pairwise similarities between all data points as defined in Equation (1.4). Simply, each pairwise similarity, denoted s_{nm}, is within the range $0 \leq s_{nm} \leq 1$: $n, m = 1, ..., N$ and since self-edges do not alter the graph Laplacian matrix, we set $s_{nn} = 0$. This similarity matrix is computed in some problem-specific manner using the self-tuning (auto-adaptive) method by [41] that utilizes a user-specified parameter representing a number of the nearest-points to each data point to be considered for calculating the dispersion parameter automatically for evaluating the Gaussian Kernel (Equation (1.5)).

Let $\boldsymbol{D}$ be the degree diagonal matrix defined in Equations (1.8) and (1.9). In our work,

we use the unnormalized Laplacian matrix $\boldsymbol{L}$ defined in Equation (1.10).

As our goal is to find a minimal cut for the Laplacian graph such that the similarities between different clusters are minimized, it is important to remind that the separation information actually starts being available from the second eigenvector and on, this is since $\lambda_1 = 0$ and its associated vector $\boldsymbol{v}_1 = (1/\sqrt{N})\mathbf{1}$ always hold, where $\mathbf{1}$ is a vector of size N such that $\mathbf{1} = (1,1,...,1)^T$.

Accordingly, the second eigenvector can be obtained as follows:

$$\boldsymbol{v}_2 = \underset{\boldsymbol{v}}{argmin} \quad \boldsymbol{v}^T\boldsymbol{L}\boldsymbol{v} = \underset{\boldsymbol{v}}{argmin} \quad \frac{1}{2}\sum_{n,m=1}^{N} s_{nm}(v_n - v_m)^2 \tag{4.1}$$

$$s.t \quad \boldsymbol{v}^T\boldsymbol{v} = 1, \quad \boldsymbol{v}^T\mathbf{1} = 0$$

For instance, by considering the embedding of data on the second eigenvector $\boldsymbol{v}_2$, we are dividing this data into two distant groups away from zero, one on the negative side and the other on the positive side. Therefore, $\boldsymbol{v}_2$ plays the role of an approximation of the two-cluster indicator vector.

4.3.2 Active Constraint Selection

We consider the Laplacian spectral properties for the selection of our cannot-link constraints. Spectral bi-partitioning, that is represented by splitting the data according to thresholding the second eigenvector $\boldsymbol{v}_2$ values, is beneficial in our case. For instance, when data is well clustered, splitting on $\boldsymbol{v}_2$ is enough to divide the dataset $\boldsymbol{X}$ into two distinct clusters. Even in the case of multi-class data the second eigenvector will still divide $\boldsymbol{X}$ into only two groups, however, this ensures that the data in group 1 should be cannot-linked to the data in group 2 even if these latter groups can be further divided themselves. Thus, regardless of the true number of classes, thresholding on $\boldsymbol{v}_2$ to split the data is sufficient for the selection of cannot-link constraints. In addition, our Active Constraint Selection (ACS) algorithm is an active iterative algorithm, which means, each time a constraint is obtained, the similarity matrix is updated accordingly, leading to a change in the Laplacian matrix. This leads to changing the second eigenvector of the graph Laplacian, which ensures that data configuration is being updated and by turn the representation of multi-class data implicitly holds. Since we are precisely interested in margin-based feature selection, and specifically in cannot-link constraints, we treat them as identifiers that not only tell that their two end points are well separated, but also tell that their nearby surrounding points are well separated

too. In our case, this local consistency hypothesis is respected and assumed to be true. Accordingly, we follow the logic of spectral analysis and benefit from the spectral properties of the graph Laplacian of data.

As we mentioned in section 4.3.1, upon embedding the data on $\boldsymbol{v_2}$, they are projected as two distant groups away from zero, one on the negative side and the other on the positive side, hence, v_2 plays the role of an approximation of the relaxed cluster indicator vector. For instance, Equation (4.1) shows that any point with approximately equal similarities $s_{nm} : n, m = 1, ..., N$ to all other points, will be embedded close to the average of all the points locations on $\boldsymbol{v}_2$, which means, it will be embedded near zero and considered an uncertain point. Nevertheless, using $\boldsymbol{v}_2$, we are not working on the real feature space, but on an equivalent projected one. Therefore, we iteratively utilize the perturbation analysis theory to actively find the couple in input space that has the greatest first order derivative with respect to this minimum point on $\boldsymbol{v}_2$ [156, 157].

Theoretically speaking, our method initially starts with a complete matrix of similarities and then mainly captures the point of minimum magnitude on the estimated $|\boldsymbol{v}_2|$ denoted $v_{2(i^*)}$ where $i^* = argmin_{i \in \{1...N\}} |v_{2(i)}|$. This point is considered the one with highest uncertainty. The method then measures the sensitivity of $v_{2(i^*)}$ to the change in s_{nm}, once it finds the similarity couple $(\boldsymbol{x}_n, \boldsymbol{x}_m)$ that is most able to change the embedding location of $v_{2(i^*)}$ on $\boldsymbol{v}_2$, only this couple is queried for a constraint and its corresponding similarity s_{nm} is amended accordingly. For instance, an external source of knowledge (e.g. an expert or a human oracle) is asked to observe the couple $(\boldsymbol{x}^*_n, \boldsymbol{x}^*_m)$ and provide an answer about whether it should be a cannot-link or a must-link constraint. If it appears that the queried couple should be a cannot-link (or must-link) constraint then s_{nm} is changed to 0 (or 1). Finally, $\boldsymbol{v}_2$ is calculated again and the process iterates until finding the required number of constraints.

Similarly to [143] and [141], we search for the couple of highest impact on $\boldsymbol{v}_2$ using:

$$(\boldsymbol{x}_{n^*}, \boldsymbol{x}_{m^*}) = \underset{(n,m)}{argmax} \left| \frac{dv_{2(i^*)}}{ds_{nm}} \right| \tag{4.2}$$

where Equation (4.2) is evaluated according to the First-Order Eigenvector Perturbation theorem [156] as follows:

$$\left| \frac{dv_{2(i^*)}}{ds_{nm}} \right| = \left| \sum_{p>2}^{N} \frac{\boldsymbol{v}_2^T [\partial \boldsymbol{L} / \partial s_{nm}] \boldsymbol{v}_p}{\lambda_2 - \lambda_p} v_{p(i^*)} \right| \tag{4.3}$$

For the sake of easy but original representation of data, we adopt the incidence vector/matrix representation [136, 158].

Definition 4.1. The incidence vector $\boldsymbol{e}_i$ is the position indicator of index i where $1 \leq i \leq N$. Thus, the i-th element of $\boldsymbol{e}_i$ is equal to 1 while the rest of the entries are 0. The length of this vector is equal to N.

Definition 4.2. We denote by $\boldsymbol{R}$ the incidence matrix holding the difference between the incidence vectors $(\boldsymbol{e}_n - \boldsymbol{e}_m)$ as its columns where $1 \leq n < m \leq N$ [158]. The incidence matrix $\boldsymbol{R}$ is an equivalent but simpler representation of our similarity matrix where each of its columns shows the similarity between two points n and m when multiplied by $\sqrt{s_{nm}}$. Thus, a similarity s_{nm} can be represented by the difference of the two incidence vectors $\left((\boldsymbol{e}_n - \boldsymbol{e}_m)\sqrt{s_{nm}}\right)$ [158]. We present below the support of this proposition, and declare that it enables the straight forward calculation of $[\partial \boldsymbol{L}/\partial s_{nm}]$.

For a Laplacian matrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{S}$ there exists an incidence matrix $\boldsymbol{R}$ by which $\boldsymbol{L} = \boldsymbol{R}\boldsymbol{R}^T$:

$$\begin{aligned} \boldsymbol{R}\boldsymbol{R}^T &= \sum_{1 \leq n < m \leq N} \left((\boldsymbol{e}_n - \boldsymbol{e}_m)\sqrt{s_{nm}}\right)\left((\boldsymbol{e}_n - \boldsymbol{e}_m)^T\sqrt{s_{nm}}\right) \\ &= \sum_{1 \leq n < m \leq N} s_{nm}(\boldsymbol{e}_n - \boldsymbol{e}_m)(\boldsymbol{e}_n - \boldsymbol{e}_m)^T \\ &= \boldsymbol{L} \end{aligned} \tag{4.4}$$

Note that,

$$(\boldsymbol{e}_n - \boldsymbol{e}_m)(\boldsymbol{e}_n - \boldsymbol{e}_m)^T = \begin{pmatrix} & \vdots & & \vdots & \\ \dots & 1 & \dots & \text{-1} & \dots \\ & \vdots & & \vdots & \\ \dots & \text{-1} & \dots & 1 & \dots \\ & \vdots & & \vdots & \end{pmatrix} \tag{4.5}$$

where

$$[\partial \boldsymbol{L}/\partial s_{nm}] = (\boldsymbol{e}_n - \boldsymbol{e}_m)(\boldsymbol{e}_n - \boldsymbol{e}_m)^T \tag{4.6}$$

This representation allows measuring the change in $\boldsymbol{L}$ with respect to the similarity change between a couple $(\boldsymbol{x}_n, \boldsymbol{x}_m)$ in $\boldsymbol{X}$. So, after substituting $[\partial \boldsymbol{L}/\partial s_{nm}]$ by $(\boldsymbol{e}_n - \boldsymbol{e}_m)(\boldsymbol{e}_n - \boldsymbol{e}_m)^T$ (applying Equation (4.6) in Equation (4.3)), we obtain the couple of highest impact on v_2 as follows:

$$(\boldsymbol{x}_{n^*}, \boldsymbol{x}_{m^*}) = \underset{(n,m)}{argmax} \left| \sum_{p>2}^{N} \frac{\boldsymbol{v}_2^T [(\boldsymbol{e}_n - \boldsymbol{e}_m)(\boldsymbol{e}_n - \boldsymbol{e}_m)^T] \boldsymbol{v}_p}{\lambda_2 - \lambda_p} v_{p(i^*)} \right| \tag{4.7}$$

Note that, since an uncertain couple is either uncertain due to having uncertainty on both of its points or due to having at least one of them considered uncertain, it is possible and common to have the point of minimum value on $\boldsymbol{v}_2$ as one of the points of the most sensitive couple $(\boldsymbol{x}_{n^*}, \boldsymbol{x}_{m^*})$. The latter can be considered with the aim of decreasing the search space to be N instead of N^2.

Finally, once we have found the couples to be queried for constraints, we apply our feature selection method presented in **Algorithm 3.2** on these constraints. It is important to note that when solving the hardest form of a specific problem, you implicitly solve the easier versions of it. Which means, selecting the most discriminative features using the most confusing constraints implicitly holds feature selection on any other true but random constraint. **Algorithm 4.1** shows the detailed steps of our constraint selection method.

Example:

To illustrate this method and highlight the significance of constraint selection, we present a toy example in Figure (4.1).

As can be seen in Figure (4.1a), the dataset is composed of 6 data points characterized by 3 features each. The scatter plot of these data points is presented in Figure (4.1b) where points 1, 2 and 3 are assigned to the first cluster and points 3, 4 and 5 are assigned to the second one. We also show the projection of these data points on the first feature A1, second feature A2 and third feature A3 in Figures (4.1c), (4.1d) and (4.1e) respectively. It is thus clear that the third feature A3 is the only one that can perfectly assign the data points to their true clusters successfully. So, it is expected from an efficient feature selection method to identify feature A3 as the most relevant one.

When we first considered an unsupervised feature selection method, i.e the Laplacian score, it failed to identify the most relevant feature A3, instead, it chose A1 which is known to be irrelevant as presented in Figure (4.1h). With the aim of improving feature selection, we used our constrained margin-based semi-supervised method presented in **Algorithm 3.2** and called ReliefF-Sc.

We first applied ReliefF-Sc on two randomly generated cannot-link constraints as presented in Figure (4.1f), but surprisingly, the ranking of features did not change and A3 was still not identified as the most relevant feature correctly. Therefore, we considered actively choosing the couples to be queried using **Algorithm 4.1** before applying ReliefF-Sc again.

Figure (4.1i) shows the change in v_2 during the process of selecting these constraints. It is divided into two steps. Step 1 shows that point 5 was the closest to zero on v_2, hence it was considered the point with the highest uncertainty, i.e. $i^* = 5$. As mentioned before, a point is uncertain when it is not clear to which cluster it should be assigned. Accordingly, after evaluating the sensitivity of $i^* = 5$ with respect to the data couples, the couple (5, 2) was chosen to be queried for a constraint. The latter was obtained from class labels in this example leading to a cannot-link constraint. The similarity matrix is updated accordingly ($s_{(5,2)}$=0) and the Laplacian matrix is recalculated obtaining a different v_2. Later, the updated v_2 appears in the second graph of (4.1i) where the closest point to zero appeared to be point 3. Thus, as can be seen in Step 2 the second constraint (3, 6) was chosen similarly to Step 1. Finally, the selected constraints can be seen in Figure (4.1g) and the last update of v_2 after finding the two constraints can be seen in the third graph of (4.1i) by which it appears that the two clusters were well separable.
The ranking of the constrained feature selection method ReliefF-Sc and the supervised method ReliefF were identical as presented in Figure (4.1h). A3 was successfully ranked as the most relevant feature, A1 the second, and the least relevant feature A2 was ranked the last as expected. Being able to reach the same ranking by ReliefF (supervised) using ReliefF-Sc (constrained) with active constraint selection (ACS) and not with random constraint generation (RCG), shows the significance of choosing constraints systematically.

One of the known drawbacks of the Relief family margin-based algorithms is that due to their initial supervised context, they need a set of points that is sufficiently big to calculate the margin [2]. However, even inexpensive oracles can turn to be resource-expensive and impose high costs when queried for a large number of constraints. Therefore, in our scenario, we should deal with a relatively small number of constraints due to the nature of the this "Active Constraint Selection (ACS)" step. Hence, in the next section, we present a method that allows the automatic propagation of these selected constraints to their unconstrained neighborhood.

Algorithm 4.1. Active Constraint Selection (ACS)

Input: Needed number of constraints $card(CL)$, $CL = \{\boldsymbol{\phi}\}$ and similarity matrix $\boldsymbol{S}$
Output: Constraint set CL

1. Initialize iter = 1
2. While iter $<= card(CL)$,
   (a) $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{S}$
   (b) Find eigenvectors $\boldsymbol{v}$ and eigenvalues λ of $\boldsymbol{L}$
   (c) $i^* = argmin_{i \in \{1 \ldots N\}} |v_{2(i)}|$
   (d) $(n^*, m^*) = argmax_{n,m \in \{1 \ldots N\}} \left| \frac{dv_{2(i^*)}}{ds_{nm}} \right|$
   (e) Query for constraints on the couple
   - if $(\boldsymbol{x}_{n^*}, \boldsymbol{x}_{m^*})$ should be a must-link constraint
     set $s_{n^*m^*} = s_{m^*n^*} = 1$
   - if $(\boldsymbol{x}_{n^*}, \boldsymbol{x}_{m^*})$ should be a cannot-link constraint
     set $s_{n^*m^*} = s_{m^*n^*} = 0$
     update CL with the couple $(\boldsymbol{x}_{n^*}, \boldsymbol{x}_{m^*})$
     iter = iter $+ 1$

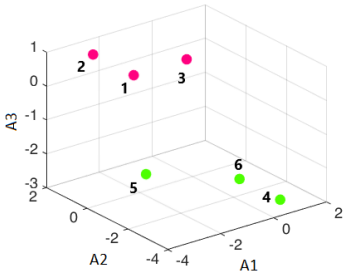
   End While

4.4 Propagation of Actively Selected Constraints

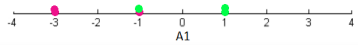
With the aim of increasing the number of available constraints for ReliefF-Sc algorithm, propagating instance-level constraints to space-level ones can be a solution [137, 144, 150, 151, 154]. The latter pieces of research tackled the problem of failing to maintain the space-level implications suggested by constraints in the clustering context; in other words, they found a mechanism for propagating the constraints to their neighborhood. However, to the best of our knowledge, no previous work has combined selecting constraints based on the graph Laplacian eigen-decomposition with

Data Point	A1	A2	A3
1	-3	-1	1
2	-3	1	1
3	-1	-1	1
4	1	-3	-3
5	-1	1	-3
6	1	-1	-3

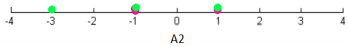
(a) Dataset



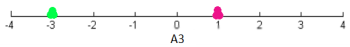
(b) 3D-scatter of data points



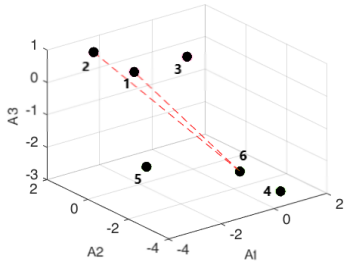
(c) Data projected on feature A1



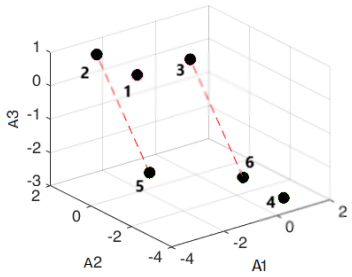
(d) Data projected on feature A2



(e) Data projected on feature A3



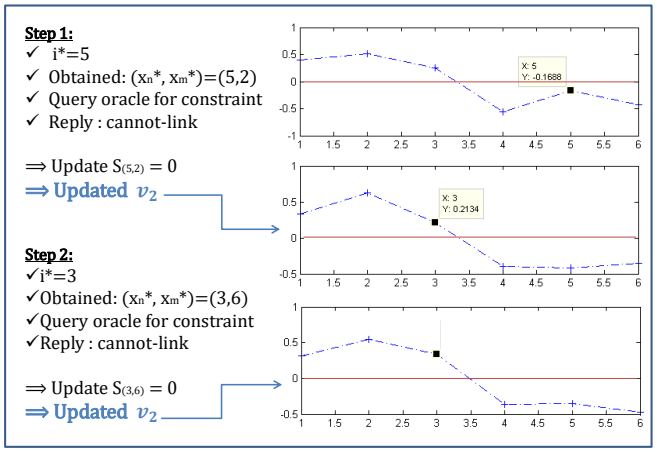
(f) Random Constraint Generation (RCG)



(g) Active Constraint Selection (ACS) by **Algorithm 4.1**

Feature Selection Method	Selected Features
Laplacian (Unsupervised)	A1,A3,A2
ReliefF-Sc + RCG	A1,A3,A2
ReliefF-Sc + ACS	A3,A1,A2
ReliefF (Supervised)	A3,A1,A2

(h) Feature ranks by different Methods



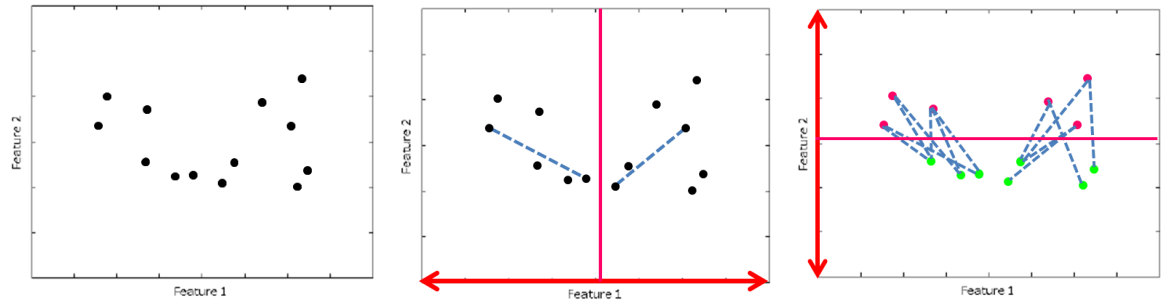
(i) Changes in v_2 during the process of Active Constraint Selection where the x-axis holds the indexes of data points and the y-axis represents the values of these data points on v_2

Figure 4.1: Three-dimensional Example Dataset: (a) Dataset, (b) 3D-scatter of data points; (c),(d),(e) Data projected on features A1, A2, A3 respectively; (i) shows changes in v_2 during ACS; (g) shows the ACS by **Algorithm 4.1**; (f) shows RCG and (h) shows the results of features ranked by different methods.

space-level constraints propagation for the sake of better feature selection.

To explain how propagation can affect feature selection, we show that while the features selected in a constrained environment are consistent with the instance-level constraints themselves, they might not be consistent with the natural global implications of these constraints. For example, in Figure (4.2) the true data partitioning is given in (4.2c) and Feature 2 should be selected, however, the two instance-level constraints shown in (4.2b) might not be enough to select this feature based on Relief algorithm, on the contrary, Feature 1 is selected. When a stronger space-level propagation occurs, it can change the results to clearly select Feature 2 as the data discriminating feature. Note that, the set of points in (4.2b) satisfies the cannot-link constraints, but Figure (4.2c), after propagation, shows a more intuitive partitioning for correct feature selection. Therefore, not only should points that are cannot-linked be in different clusters, but also their neighboring ones.

We thus explain the method we adapted to our scenario for propagating constraints to the neighboring points surrounding our previously obtained ones using **Algorithm 4.1**. It is intuitive to assume that if two points are cannot-linked, their nearest neighbors should also be cannot-linked due to neighboring point similarity rule. One main obstacle known about propagating cannot-link constraints is that they are not transitive. On the contrary to working with a must-link propagation, if x_m, x_n are cannot-



(a) Data (b) Instance-level cannot-link constraints (c) Propagated information

Figure 4.2: The effects of propagating cannot-link constraints to the data in (a): knowing that the true partitioning is shown in (c) and that Feature 2 should be selected, two instance-level constraints only may be not enough to select Feature 2 based on Relief family distance-based feature selection (b); Whereas, a stronger space-level propagation changes results to clearly select Feature 2 as the data discriminative feature (c).

linked and $\boldsymbol{x}_n$, $\boldsymbol{x}_j$ are cannot-linked we can't derive the conclusion that $\boldsymbol{x}_m$, $\boldsymbol{x}_j$ are also cannot-linked. Therefore, we decide to generalize the cannot-link constraints that we initially have to include neighboring points. This is done through an iterative procedure that is not limited to two-class problems, allows soft constraints, and converges to a closed form solution with computational efficiency.
To increase the number of pairwise constraints with the aim of exerting a stronger effect on the constrained learning process, in [148], they presented an exhaustive and efficient method for constraints propagation. Since this propagation method is usually applied as a pre-step independently from clustering, it is possible to use it before any other constrained learning method to be enhanced; just like filter-type methods in feature selection.

Definition 4.3. We denote by $\boldsymbol{Q} = \{\boldsymbol{Q}_{nm}\}_{N\times N}$ the initial indicator sparse matrix of cannot-link constraints obtained by **Algorithm 4.1** as:

$$\boldsymbol{Q}_{nm} = \begin{cases} 1, & \text{if } (\boldsymbol{x}_n, \boldsymbol{x}_m) \in CL \\ 0, & \text{otherwise} \end{cases} \tag{4.8}$$

Note that, in this paper, we are interested in propagating our actively chosen cannot-link pairwise constraints in order to provide more information to Relief-Sc for applying margin-based feature selection, this is why the indicator matrix $\boldsymbol{Q}_{nm}$ holds only cannot-link constraints.

Although the method by [152] is capable of applying vertical constraint propagation, i.e along the columns of the initial constraint matrix $\boldsymbol{Q}$, sometimes a column might not contain any pairwise constraint, this means that all the cells of this column might have the value of zero and therefore no vertical constraint propagation can be obtained for this particular column. Thus, [148] not only modified the label propagation method to a constraint propagation one, but also solved empty constraint column problem by two exhaustive and efficient vertical and horizontal propagation steps. The latter step is usually applied after finishing vertical propagation in an iterative approach, and leads to a full matrix that contains a constraint between every two data points in $\boldsymbol{X}$.

Similarly to [148] and [153], we use the normalized similarity graph to propagate the constraints in a data-specific manner. For that, we denote by $\{\boldsymbol{P}_{nm}\}_{N\times N}$ the neigh-

borhood matrix calculated as follows:

$$\boldsymbol{P} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{S}\boldsymbol{D}^{-\frac{1}{2}} \tag{4.9}$$

where $\boldsymbol{P}$ is a symmetric matrix having real valued entries between data points. Note that, unlike [148] we use a fully connected symmetric similarity graph instead of the K-NN graph to build $\boldsymbol{S}$.

Definition 4.4. Let $\boldsymbol{G} = \{\boldsymbol{G}_{nm}\}_{N\times N} : |\boldsymbol{G}_{nm}| \leq 1$ be the matrix that holds the propagated constraints at each iteration of the algorithm while retaining initial information from $\boldsymbol{Q}$.
The t-th iteration for a vertical propagation is given by:

$$\boldsymbol{G}_v(t+1) = a\boldsymbol{P}\boldsymbol{G}_v(t) + (1-a)\boldsymbol{Q} \tag{4.10}$$

And it converges to [148]:

$$\boldsymbol{G}_v^* = (1-a)(I - a\boldsymbol{P})^{-1}\boldsymbol{Q} \tag{4.11}$$

The t-th iteration for a horizontal propagation is given by:

$$\boldsymbol{G}_h(t+1) = a\boldsymbol{G}_h(t)\boldsymbol{P} + (1-a)\boldsymbol{G}_v^* \tag{4.12}$$

Similarly, it converges to [148]:

$$\boldsymbol{G}_h^* = (1-a)\boldsymbol{G}_v^*(I - a\boldsymbol{P}^T)^{-1} \tag{4.13}$$

where a is a regularization parameter between $[0,1]$, it regulates the amount of information a data point receives from its neighbors graph $\boldsymbol{P}$ and its initial state $\boldsymbol{Q}$. It was initially set to 0.99 by [153] The convergence analyses of vertical and horizontal propagations are stated by [153] and [148] respectively.
Finally, from Equations (4.11) and (4.13) the over all closed-form solution can be calculated in a quadratic time $O(kN^2)$ as:

$$\begin{aligned}\boldsymbol{G}^* &= \boldsymbol{G}_h^* \\ &= (1-a)^2(I - a\boldsymbol{P})^{-1}\boldsymbol{Q}(I - a\boldsymbol{P})^{-1}\end{aligned} \tag{4.14}$$

At the end of this propagation method a full matrix of soft pairwise constraints is obtained. Lu and Ip dealt with these values as confidence scores that are thresholded upon zero leading to the adjustment of their similarity matrix accordingly. However,

since our aim is not constrained spectral clustering but constrained feature selection, we need to find an appropriate threshold τ to filter the propagated constraints. Accordingly, any couple corresponding to a cell in $\boldsymbol{G}^*$ that has a value less than τ will be disregarded and any couple corresponding to a cell in $\boldsymbol{G}^*$ that has a value greater than or equal to τ will be considered a cannot-link constraint in the new set of propagated constraints CL^*. Thus, we calculate the threshold τ as the mean of the maximum values obtained from each row of $\boldsymbol{G}^*$.
Then,

$$CL^* = \left\{ \ (\boldsymbol{x}_n, \boldsymbol{x}_m) \quad \text{if} \ \ \boldsymbol{G}^*_{nm} \geq \tau \ \right\} \tag{4.15}$$

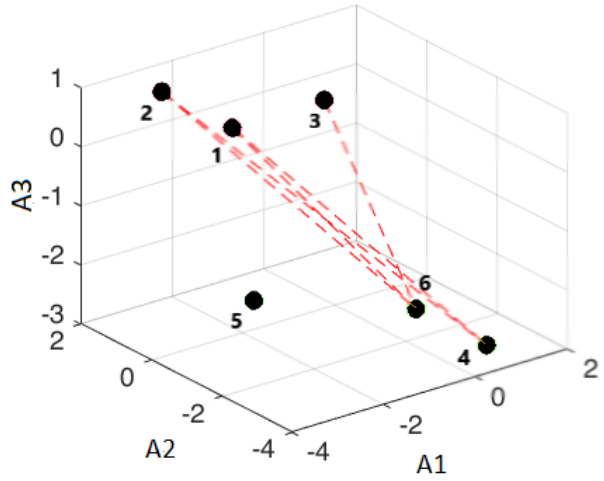
We show our overall method step by step to reach constraint propagation in **Algorithm 4.2**. Note that the below standalone constraint propagation algorithm can be applied prior to feature selection independently from any feature selection algorithm.

Algorithm 4.2. Standalone Constraint Propagation

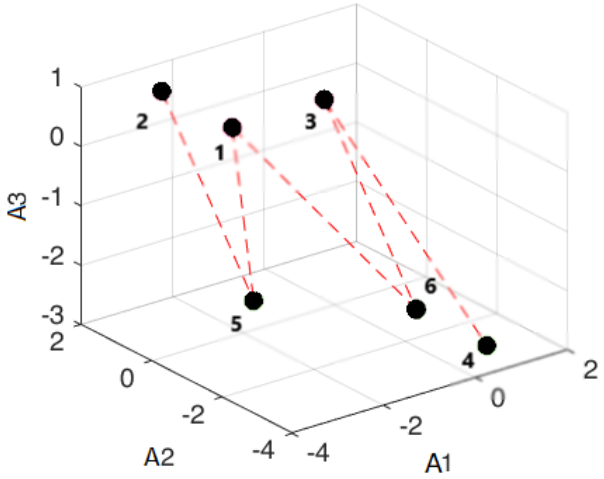
Input: Actively Chosen Constraint set CL by **Algorithm 4.2**
Output: Updated Constraint set CL^* by Propagated constraints

1. Initialize the constraint matrix $\boldsymbol{Q}$ from CL using (4.8)
2. Calculate the normalized similarity graph $\boldsymbol{P}$ using (4.9)
3. Apply the Vertical constraint propagation using (4.11)
4. Apply the Horizontal constraint propagation using (4.13)
5. Calculate the threshold $\tau = Mean\ (Maximum\ of\ each\ row\ in\ \boldsymbol{G}^*)$
6. Obtain CL^* using (4.15)

Finally, Figure (4.3) shows the results of propagating the initial constraints observed in Figure (4.1) using **Algorithm 4.2**. As expected, ReliefF-Sc with PACS just maintained the correct rank of features since ReliefF-Sc with only ACS was also able to correctly rank them. However, Figure (4.3c) shows that the false rank by ReliefF-Sc with RCG was fortunately corrected upon applying propagation (PRCG). This is due to augmenting supervision information although point 5 was left without a constraint as can be seen in Figure (4.3a).



(a) Random constraints after propagation (PRCG)



(b) Actively selected constraints after propagation (PACS)

Feature Selection Method	Selected Features
ReliefF-Sc + Random Constraints	A3,A1,A2
ReliefF-Sc + Constraints Selection	A3,A1,A2

(c) Results of ReliefF-Sc after propagation with and without Active Constraint Selection.

Figure 4.3: The resulting propagation of our previous actively selected constraints: (a) Propagation of initial random constraints (6, 2) and (6, 1), (b) Propagation of actively selected constraints (5, 2) and (3, 6) by **Algorithm 4.1**, and (c) Improved results of feature selection after propagation of constraints.

4.5 Complexity Analysis

As for computational complexity of ReliefF-Sc, it is calculated as mentioned in the previous chapter, $O(card(CL) \times NF)$, knowing that $card(CL)$ is the number of cannot-link constraints, N is the number of data points and F is the dimension of feature space. Under the ACS framework, the computational complexity by ReliefF-Sc is not the problem since we aim to actively select as few as possible good constraints. However, this selection of constraints using **Algorithm 4.1** can turn to have a very high-cost overhead when the number of data points N is relatively large. This is due to the necessity of calculating Equation (4.7) each time we want to find the sensitivity matrix. The latter holds all the sensitivities of all data couples to the minimum point i^* on $\boldsymbol{v_2}$. Thus, Equation (4.7) that costs $O(N^3)$ has to be repeated as long as we still need more constraints. Therefore, the overall computational complexity of **Algorithm 4.1** is calculated as $O(card(CL) \times N^3)$. This is an issue to most of the pair-based constraints selection algorithms due to the nature of the pair selection problem that needs to rank $O(N^2)$ candidate pairs during each of its searching iterations [136].

4.6 Experimental Results

We first describe the experimental setup, where we aim at comparing the classification accuracy obtained by the different supervised, unsupervised and semi-supervised constrained feature selection algorithms listed in section 4.6.1. We specify the used datasets in section 4.6.2 and the performance evaluation measures in section 4.6.3.

In our experiments, we seek to show that our framework with active constraint selection replacing random constraint generation can significantly improve constrained feature selection algorithms especially our margin-based ReliefF-Sc. Note that, ACS is algorithm-independent, which means that unlike other constraint selection methods [138, 139, 147], our method can be used to actively choose constraints regardless of the used constrained feature selection method. We also show the significance of propagating constraints and its effect on the classification accuracy curves. The comparison results are detailed in section 4.6.4.

4.6.1 Used Benchmarking Feature Selection Methods

To evaluate our feature selection framework using the proposed active constraint selection and propagation methods, we benchmark with the following well-known feature selection methods including our own: Fisher score, ReliefF, Laplacian score, Con-

straint score-1 (CS1), Constraint score-2 (CS2) (where λ is set to the default value given by authors i.e 0.1 [40]), Simba-Sc (Start points parameter is set to its default value i.e 5 [107]), Constraint score-4 (CS4), Constrained Laplacian score (CLS), and ReliefF-Sc the proposed semi-supervised margin-based constrained feature selection method detailed in section 3.2.3.
In the following, ReliefF-Sc is applied in three versions: (1) alone with RCG denoted ReliefF-Sc+RCG, (2) after applying ACS presented in **Algorithm 4.1** denoted ReliefF-Sc+ACS, and (3) as a combination of **Algorithms 3.2**, **4.1** and **4.2** that is detailed in section 4.4. The latter is denoted as ReliefF-Sc+PACS.

4.6.2 Datasets Description and Parameter Setting

We evaluate our proposed active constraint selection and propagation framework on five well-known numeric UCI machine learning datasets [3], including Soybean, Wine, Heart, Sonar and Image Segmentation. We also test on another two high dimensional gene-expression datasets: ColonCancer [131] and Leukemia [132]. Note that, our experiments are divided according to their goal into two parts. The first part, including sub-sections 4.6.4.1 and 4.6.4.2, focuses on highlighting the effect of Active Constraint Selection in comparison to Random Constraint Generation and Constraint Propagation and is applied on Soybean, Wine, Heart, Sonar, ColonCancer and Leukemia datasets using the 1-NN classifier. The second part, including sub-sections 4.6.4.3 and 4.6.4.4, focuses on the comparison with other feature selection methods and is applied on Wine, Image Segmentation and ColonCancer using the 1-NN and SVM classifiers. Image Segmentation dataset was used in the second part in order to validate the results over a dataset with a greater number of classes.

Table 4.1 summarizes the main characteristics of each dataset mentioned in the first column. The second, third, and fourth columns show the number of data points, the data dimension, and the number of classes respectively. Also, the fifth column specifies the number of points in each class in order to allow observing whether the classes are balanced or not and the last column indicates the number of queried constraints. Before recording the results, and for the sake of a fair comparison between different methods, we normalize all features to be between zero and one. Also, the similarity between data points is computed in some problem-specific manner, generally, we use the self-tuning (auto-adaptive) method by [41] that takes only a user-specified parameter representing the number of the nearest points to each data point and calculates the dispersion parameter automatically for evaluating the Gaussian Kernel.

Table 4.1: Datasets and their main characteristics

Dataset	#data points	#features	#classes	#points per class	#constraints
Soybean	47	35	4	10, 10, 10, 17	3
Wine	178	13	3	59, 71, 48	20
Heart	270	13	2	150, 120	20
Sonar	208	60	2	97, 111	10
Segmentation	210	19	7	30, 30, 30, 30, 30, 30, 30	10
ColonCancer	62	2000	2	40, 22	8
Leukemia	72	5147	2	47, 25	10

In these experiments, we divide our datasets into training and testing sets defined by the same features, the first half of data points from each class is considered as the training set and the second half is considered as the testing set. We were also careful that no testing data is seen by either the feature selection algorithm or the classifier during the learning process.

Regarding constraints, it is important to know that they can be generated from background knowledge given by an expert, i.e querying an oracle, or acquired experimentally from a set of labeled data [138]. In our experiments, since we compare the effect of random constraints to the effect of constraint selection, we average the results over 100 runs for the random constraint sets. At each run, random constraints are generated as follows: a pair of data points is randomly picked from the training set, then its class assignment is checked if it turns to be that these two points belong to different classes they are considered a cannot-link constraint couple. This operation is repeated until the desired number of constraints is met. While for actively selected constraints, we select them using **Algorithm 4.1** where the user only provides information about the couple considered most uncertain. Since cannot-link constraints are considered more important than must-link ones from the viewpoint of margin, **Algorithm 3.2** (ReliefF-Sc) utilizes them only. Note that, we set the same value of K (number of nearest neighbors) for both ReliefF and ReliefF-Sc on each dataset according to its characteristics. This value was set to 5 for Wine, Heart, Sonar and Leukemia and 3 for Soybean and Segmentation, however, only for ColonCancer we kept it equal to 1 since we experimentally noticed that this value provided the highest accuracy curve for this particular dataset.

Also, our criteria to choose the number of pairwise constraints to be queried is as follows: we check the approximate number of points around zero on the second eigenvector v_2, and we consider it as the minimum supervision information needed. Hence,

we use the number of cannot-link constraints specified by [107] for the datasets Wine, Heart, and Sonar i.e 20, 20 and 10 respectively since they fit our criteria, we also use 10 constraints for Image Segmentation similarly to [53], however, for Soybean, Colon-Cancer and Leukemia datasets we assign 3, 8 and 10 constraints respectively. Note that, for the algorithms that use both must-link and cannot-link constraints, for example, CS1 and CS2, we provide half the number of constraints as cannot-link and half as must-link (chosen in the same manner when using ACS or RCG). This is to ensure providing the same amount of supervision information to all algorithms.

4.6.3 Performance Evaluation Measures

After applying feature selection, we evaluate its performance using three evaluation measures: one is related to the data classification accuracy obtained by applying a classifier on the selected set of features, and two others are directly related to measuring the quality of the selected feature set, i.e., Precision and Distance measures [159].

1. Classification Accuracy: is defined as a percentage of correct predictions. Thus, we evaluate how many data points were correctly classified using the chosen feature subset. It is defined in Equation (1.18) of section 1.5.4.1 as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

   In our experiments, we evaluate the classification accuracy of each ranked set of features by applying two well known classifiers: the Nearest Neighbor (1-NN) classifier with euclidean distance and the SVM classifier with linear kernel function (default configurations in Matlab-stats toolbox are used). For this purpose, first the feature selection is applied on the training set, this provides the ranking of features according to their assigned scores by different algorithms. Then, classification using the selected features is applied on the testing set.

2. Precision (Pr) [159]: is here calculated as the number of features present in two lists: Target T_r that represents the first s features present in the optimal feature ranking list S_{opt} and R_s that represents the first s features found in a selected set of features resulting from the feature selection method, divided by the number of features in T_r denoted by $card(T_r)$ (i.e $card(T_r) = s$):

$$Pr = \frac{card(T_r \cap R_s)}{card(T_r)} \tag{4.16}$$

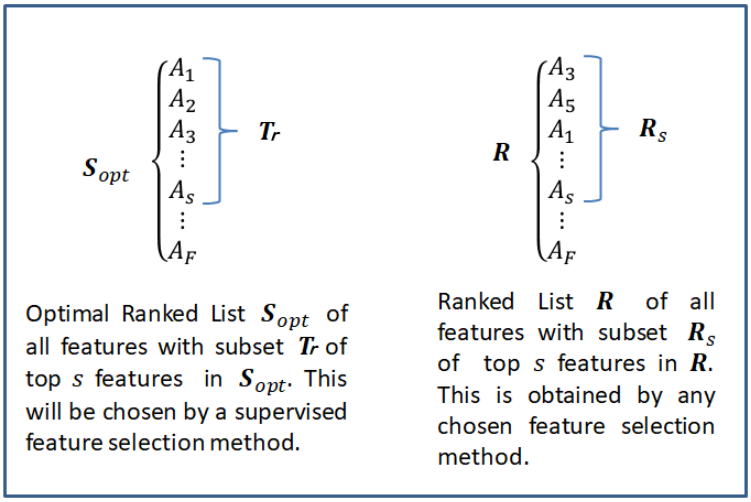


Figure 4.4: Relationship between the terms used in the Performance measures: Precision and Distance.

This measure is between zero and one, knowing that $Pr = 1$ when the subsets T_r and R_s contain the same features and $Pr = 0$, when there is no intersection between them. The higher precision the better result. For clarity we present the relations between all terms in Figure (4.4). Note that, Precision (Pr) here is different from the one presented in Equation (1.19).

3. Distance [159]: since Precision cannot consider the order of features between T_r and R_s, we also use Distance Dt. This measure is defined based on the sum of distances between the intersecting features of T_r and R_s. We calculate the distance of each feature by comparing its position in T_r and R_s.
Let us consider S'_{opt} the reverse of S_{opt} to calculate the maximum possible distance Dt_{max} between two lists having the same features as:

$$Dt_{max} = \sum_{\forall A_i \in S_{opt}} |position(A_i \in S_{opt}) - position(A_i \in S'_{opt})| \tag{4.17}$$

Where Dt_{max} is used to keep distance Dt between zero and one, knowing that Dt is at its best when it is zero with the two lists having identical ranking and larger than zero otherwise.

Finally, Dt is calculated as follows:

$$Dt = \frac{\sum_{\forall A_i \in T_r} |position(A_i \in T_r) - position(A_i \in R)|}{Dt_{max}} \tag{4.18}$$

4.6.4 Performance Evaluation Results

In this section, we first compare the classification accuracy obtained after applying **Algorithm 3.2** (ReliefF-Sc with RCG) to the classification accuracy obtained after applying the same algorithm but with active constraint selection (ReliefF-Sc with ACS) in section 4.6.4.1. We then propagate the selected constraints and compare the classification accuracy obtained after applying ReliefF-Sc on these constraints to the classification accuracy obtained without propagation in section 4.6.4.2. Then, in sections 4.6.4.3 and 4.6.4.4, we compare supervised (Fisher, ReliefF), unsupervised (Laplacian), and semi-supervised constrained feature selection algorithms (ReliefF-Sc, Simba-Sc, CS1, CS2, CS4, CLS) also with RCG and ACS.

4.6.4.1 ReliefF-Sc with Random Constraint Generation (RCG) vs. ReliefF-Sc with Active Constraint Selection (ACS)

Figures (4.5a–4.5f) show the plot of classification accuracies vs. the desired number of features ranked by ReliefF-Sc on Soybean, Wine, Heart, Sonar, ColonCancer, and Leukemia respectively. We also record the highest accuracy rates Acc and the corresponding dimension d of the selected feature subset obtained by ReliefF-Sc with and without constraint selection in Table 4.2.

As can be seen in Figure (4.5), it is generally always the case that the classification accuracy while using the constraints chosen by **Algorithm 4.1** is better than the accuracy using the same number of random constraints. For instance, Figures (4.5a) and (4.5b) show that with constraints selection we were able to reach the optimal classification of data points using only a few selected features. Moreover, it can be noticed that generally, ReliefF-Sc with ACS obtains higher classification accuracies from the first ranked feature as can be seen in Figures (4.5b), (4.5d), (4.5e), and (4.5f), which means that applying feature selection on actively selected constraints is able to enhance feature selection and identify the most discriminative feature earlier.

Note that, the results of ReliefF-Sc with RCG are averaged over 100 runs, which means, their results implicitly hold more information, this masks the steep curve fluctuations through accuracy-averaging. Whereas ReliefF-Sc with ACS is deterministic and only applied once, this is due to the closed-form solutions obtained by both **Algorithm 4.1** and ReliefF-Sc. In fact, we get the same selected constraints and the same selected features regardless of the number of times we repeat a run. Moreover, note that the applicability of a proposed method is a very important issue. For instance, applying feature selection algorithms using randomly selected constraints might some-

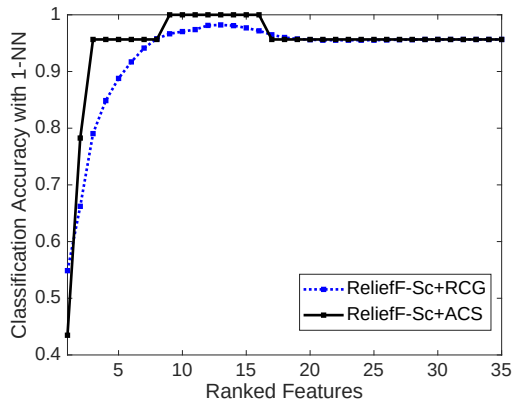
times obtain higher classification accuracies, but it is not logical nor feasible to query for the required number of random constraints in some real-world situations (e.g. 8 constraints for ColonCancer requested for 100 times in order to average their results), it might then be less expensive to label the data than to query for 800 constraints. This issue reduces the applicability of RCG although it is usually used.

Moreover, from Table 4.2 we can see that feature selection by ReliefF-Sc with ACS on Wine and Soybean boosted classification accuracy to reach 100%, which means that with actively selected constraints, ReliefF-Sc was able to successfully select the features that can provide an optimal classification of data by choosing only 9 features out of 35 on Soybean and 5 features out of 13 on Wine, unlike the results of RCG that required choosing 4 additional features in order to reach its highest accuracy rate i.e. 98.22% on Soybean and choosing 6 additional features to reach 97.75% on Wine. In addition, although the result on Sonar presented in Figure (4.5d) was problematic, this table also shows that ReliefF-Sc with ACS was still able to enhance the highest possible accuracy rate to become 62.50% on 29 which is 4.85% higher with a dimension that is smaller by 1 feature.

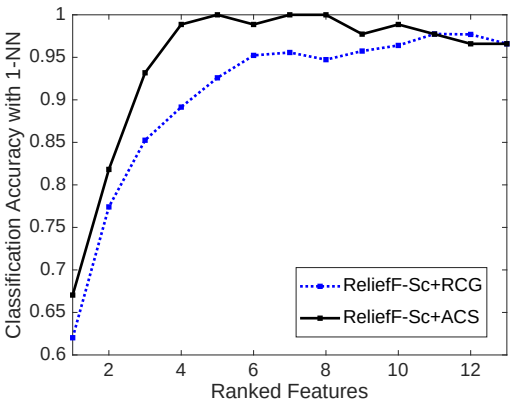
Furthermore, we insist on the impressive result obtained on the high-dimensional gene expression ColonCancer dataset, where ReliefF-Sc with selected constraints was able to reach its highest accuracy from the first chosen feature, this shows that the algorithm was able to directly decide the most significant feature in the dataset using our constraints. Whereas, it had to choose 1316 features to reach its highest accuracy using random constraints. Finally, for Leukemia dataset, Table 4.2 shows that after selecting only 198 features out of 5147 the highest classification accuracy reached 91.18% using ACS, which is 7.74% higher than the accuracy using RCG i.e 83.44% with 388 selected features.

Therefore, the significance of choosing the data couples to be queried for constraints and then to be used for feature selection is here verified. Also, as the accuracy rates achieved without feature selection on the original feature space were always lower than the accuracy rates achieved with feature selection, thus the importance of feature selection in improving the classification results is also confirmed.

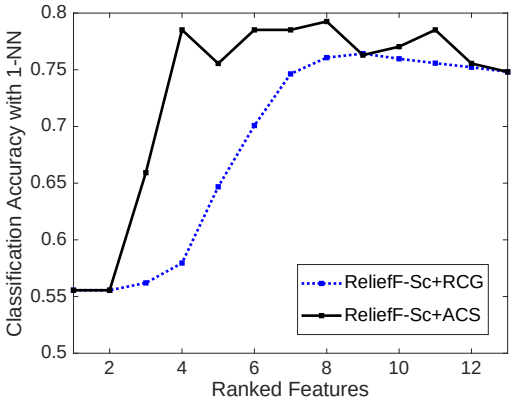
As we mentioned before, one of the known drawbacks of the Relief family algorithms is that they might need a sufficiently big set of points to calculate the margin. Hence, since we care for asking the oracle as few questions as possible and for maintaining feasibility, we might sometimes obtain a very small number of constraints. As a



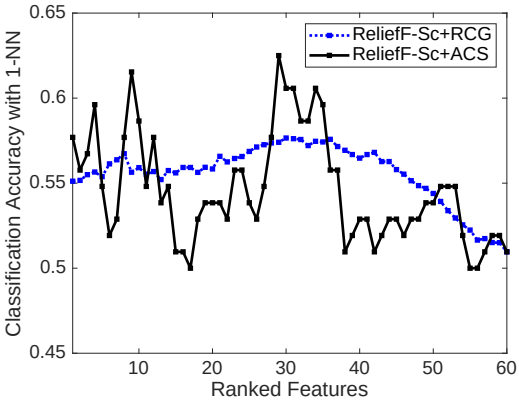
(a) ReliefF-Sc with RCG and ACS on Soybean



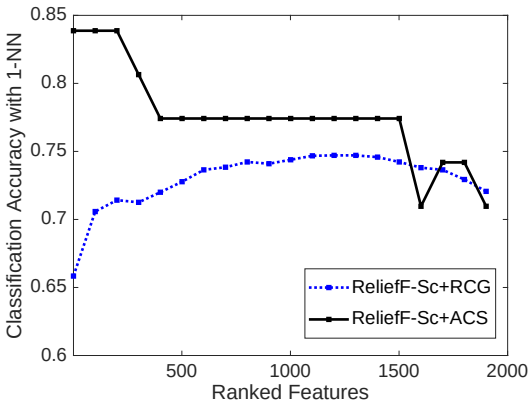
(b) ReliefF-Sc with RCG and ACS on Wine



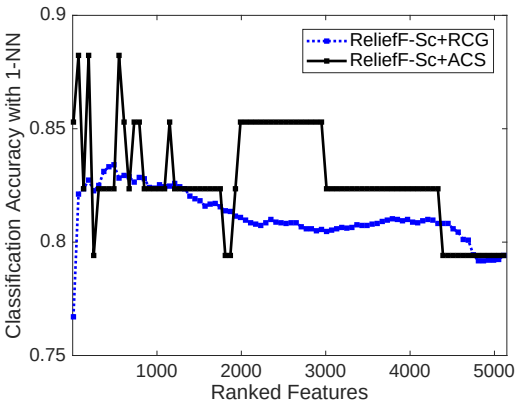
(c) ReliefF-Sc with RCG and ACS on Heart



(d) ReliefF-Sc with RCG and ACS on Sonar



(e) ReliefF-Sc with RCG and ACS on Colon-Cancer



(f) ReliefF-Sc with RCG and ACS on Leukemia

Figure 4.5: Accuracy rates using 1-NN classifier vs. different number of selected features obtained by ReliefF-Sc+RCG and ReliefF-Sc+ACS on 4 UCI datasets: (a) Soybean; (b) Wine; (c) Heart; (d) Sonar; and on 2 high dimensional gene-expression datasets: (e) ColonCancer; (f) Leukemia.

Table 4.2: The highest classification accuracy rates *Acc* (in %) obtained using 1-NN classifier on the features ranked by ReliefF-Sc algorithm with RCG and ACS, where *d* represents the dimension by which *Acc* is reached and *F* represents the original feature space.

	Without Selection		ReliefF-Sc+RCG		ReliefF-Sc+ACS	
Dataset	*Acc*	*F*	*Acc*	*d*	*Acc*	*d*
Soybean	95.65	35	98.22	13	**100**	**9**
Wine	96.59	13	97.75	11	**100**	**5**
Heart	74.81	13	76.42	9	**79.26**	**8**
Sonar	50.96	60	57.65	30	**62.50**	**29**
ColonCancer	70.97	2000	74.87	1316	**83.87**	**1**
Leukemia	79.41	5147	83.44	388	**91.18**	**198**

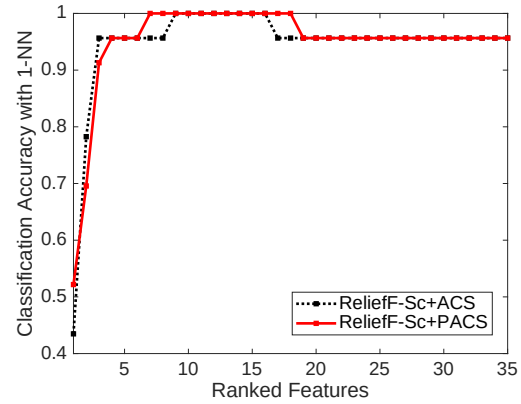
solution, in the next section, we experiment propagating constraints through transmitting supervision information to the unlabeled and unconstrained remaining nearby data points.

4.6.4.2 Results by ReliefF-Sc after the Propagation of the Actively Selected Constraints (PACS)

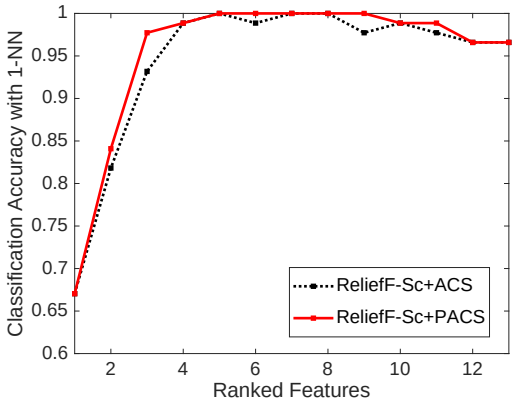
We plot the classification accuracies with PACS in Figure (4.6) and we record the highest accuracy rates in Table 4.3 before and after propagation. In addition, we show the sensitivity analysis of the propagation process to the regularization parameter in Figure (4.7).

Figure (4.6) shows that generally the accuracy curves with PACS were higher than they were with ACS only. Also, PACS generally either maintains the same accuracy levels or obtains better ones like on Wine, Heart and ColonCancer presented in Figures (4.6b), (4.6c) and (4.6e) respectively.

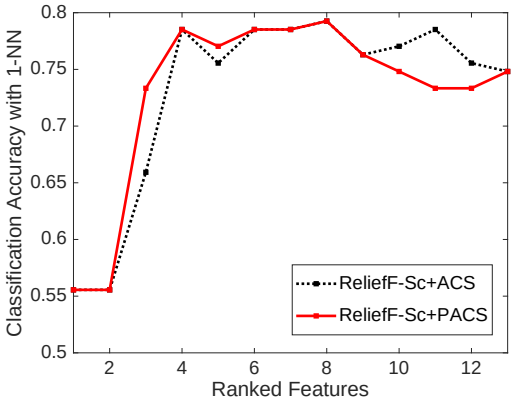
Since we already obtained the highest possible accuracy rate i.e 100% on Soybean and Wine using ACS, Figures (4.6a) and (4.6b) show that propagation enabled maintaining this accuracy on a wider span of selected features. The rows corresponding to Soybean and Wine in Table 4.3 also record the series of these selected features. For instance, this high accuracy was maintained on a span of 5 features (i.e 5,6,7,8,9) after propagation compared to 3 features (i.e 5,7,8) before propagation on Wine. Also, for Soybean a span of 12 features (i.e 7,8,9,10,11,12,13,14,15,16,17,18) was maintained after propagation compared to 8 features (i.e 9,10,11,12,13,14,15,16) before propaga-



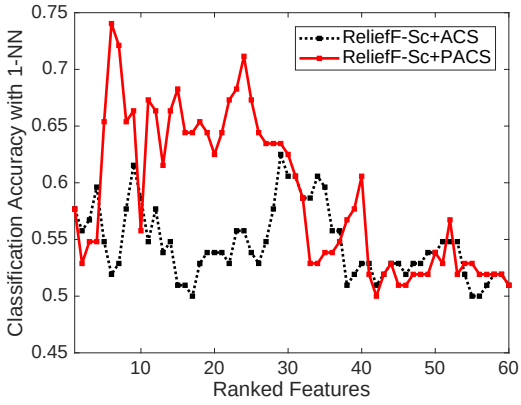
(a) ReliefF-Sc with PACS on Soybean



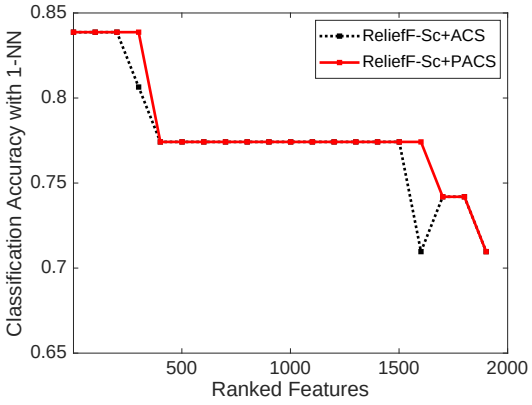
(b) ReliefF-Sc with PACS on Wine



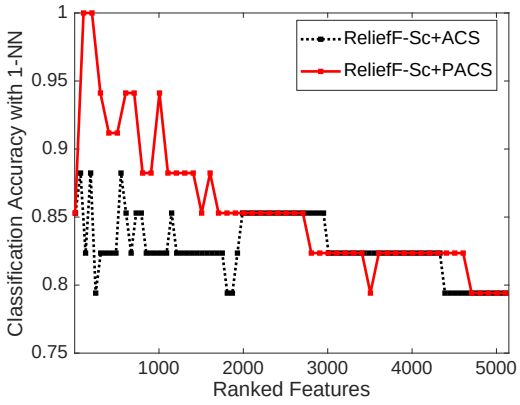
(c) ReliefF-Sc with PACS on Heart



(d) ReliefF-Sc with PACS on Sonar



(e) ReliefF-Sc with PACS on ColonCancer



(f) ReliefF-Sc with PACS on Leukemia

Figure 4.6: Accuracy rates using 1-NN classifier vs. different number of selected features obtained by ReliefF-Sc+ACS and ReliefF-Sc+PACS on 4 UCI datasets: (a) Soybean; (b) Wine; (c) Heart; (d) Sonar; and 2 high-dimensional gene-expression datasets: (e) ColonCancer; (f) Leukemia.

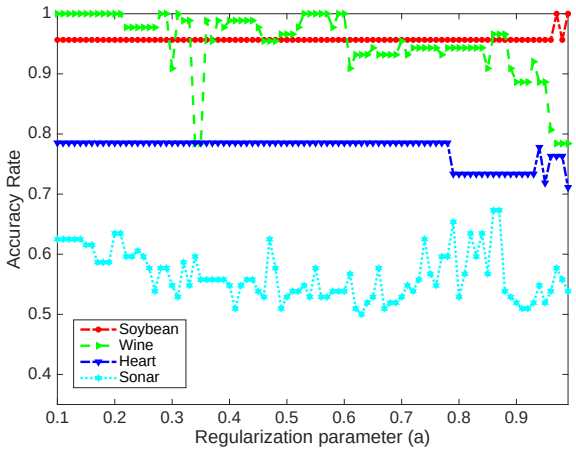


Figure 4.7: Accuracy rates over the first 50% ranked features using 1-NN classifier vs. different values of the regularization Parameter a ranging in $]0,1[$.

tion. It is also worth the note that the problematic result on Sonar was significantly enhanced obtaining a much higher accuracy curve as can be seen in Figure (4.6d), which ensures a better feature selection process. Moreover, ReliefF-Sc with PACS on Leukemia was able to obtain 100% of accuracy from the first few ranked features as can be seen in (4.6f) although with ACS (that was in its turn significantly higher than the accuracy using RCG) it was able to reach 91.18% maximum. Table 4.3 insists on the latter. For instance, ReliefF-Sc with PACS on Leukemia reached this highest accuracy on only 74 features after it had to choose 198 with ACS alone. Also, it shows that the highest classification accuracy on Sonar increased from 62.50% before constraints propagation to 74.07% with only 6 selected features out of 60 in the original space.

Note that, the regularization parameter $a = 0.99$ ([153]) can be experimentally tuned to better fit each dataset. For Soybean, we keep $a = 0.99$. However, we experimentally set a to the value of 0.6 for Wine, 0.37 for Heart, 0.47 for Sonar, 0.46 for ColonCancer, and 0.75 for Leukemia dataset. In order to analyze the sensitivity of the constraints propagation process to the regularization parameter a, we present Figure (4.7). It shows the fluctuation in accuracy on the first 50% ranked features when feature selection is applied on propagated constraints. For example, Sonar dataset has 60 features in the original feature space, we show the change in accuracy on exactly the first 30 selected features upon using all the possible values of a ranging between 0 and 1 exclusively.

This Figure shows that for Soybean and Heart datasets the obtained sets of propagated constraints are less sensitive to the regularization parameter than those of Wine

and Sonar. In fact, since Soybean already has a very small number of training data points (i.e. 24 points) the effect of constraints remained the same on feature selection until the regularization parameter reached the last few values (that are greater than 0.9). This means that more information is considered from the neighborhood graph and less is considered from the initial constraint matrix $\boldsymbol{Q}$, which allows more propagation to the neighborhood leading to a considerable number of propagated constraints with respect to the size of data (23 constraints on 24 data points). However, due to the noisy Sonar data, it was harder to follow the behavior with respect to a on it as the fluctuation was generally over all the analysis curve. It was also interesting as we noticed that using a value of 0.47 the number of propagated constraints reached 102 remembering that we have only 104 training data points and noting that the initial set of actively chosen constraints before propagation had only 10 constraints. Hence, for Sonar we decided to use $a = 0.47$.

In conclusion, we do not expect the propagation to be independent of its only parameter a; However, Figure (4.7) shows a great deal of stability even while a big change appears in the value of a (like moving from $a = 0.1$ to $a = 0.7$ on Heart). There is always some critical value of a by which the accuracy differs obviously (like $a = 0.78$ on Heart and $a = 0.6$ on Wine) except for Sonar. Therefore, we can analyze the data, if its similarity graph possess some kind of structure to the data by itself, then we can increase the value of a to consider more information from the neighborhood graph and allow higher levels of propagation. On the other side, when no clear structure of the data is provided by the similarity graph, then we keep the value of a in small ranges in order to consider more information from the matrix of chosen constraints and to keep low levels of propagations (propagate only to very close neighboring points).

Table 4.3: The highest classification accuracy rates Acc (in %) obtained using 1-NN classifier on the features ranked by ReliefF-Sc before and after constraints propagation, with d representing the dimension where Acc is reached. Initial constraints before propagation obtained systematically using **Algorithm 4.1**.

	Without Selection		ReliefF-Sc+ACS		ReliefF-Sc+PACS	
Dataset	*Acc*	*F*	*Acc*	*d*	*Acc*	*d*
Soybean	95.65	35	100	9 –> 16	**100**	**7 –> 18**
Wine	96.59	13	100	5, 7, 8	**100**	**5 –> 9**
Heart	74.81	13	**79.26**	**8**	**79.26**	**8**
Sonar	50.96	60	62.50	29	**74.07**	**6**
ColonCancer	70.97	2000	**83.87**	**1**	**83.87**	**1**
Leukemia	79.41	5147	91.18	198	**100**	**74**

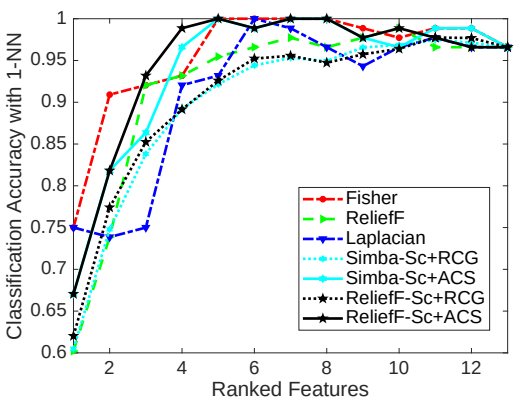
After detailing the performance results using ReliefF-Sc alone with RCG, ACS and PACS, in the next section, we wish to compare the algorithms detailed in section 4.6.1 with RCG and ACS on Wine, Segmentation and ColonCancer datasets using both 1-NN and SVM classifiers.

4.6.4.3 Classification Accuracies by Different Constrained Feature Selection Methods with Random Constraints Generation (RCG) vs. with Active Constraints Selection (ACS)

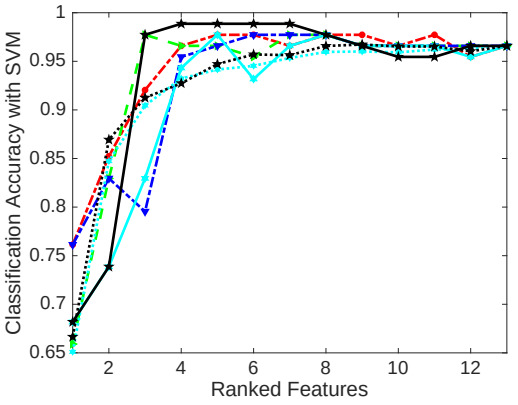
This section aims at benchmarking the different supervised, unsupervised, and semi-supervised constrained feature selection algorithms. It also aims at showing that the process of active constraint selection is independent of the constrained feature selection algorithm and can enhance the chosen set of features regardless of the algorithm to be used. For that, we plot the accuracy curves obtained by 1-NN and SVM classifiers after feature selection using Fisher, ReliefF, Laplacian and the constrained margin-based ReliefF-Sc and Simba-Sc algorithms on Wine, Segmentation and ColonCancer in Figure (4.8). For these latter accuracies, Table 4.4 further presents the highest accuracy rates Acc and their corresponding dimensions d obtained by all the algorithms detailed in section 4.6.1 with and without constraint selection.

Moreover, in Figure (4.9) of this section, we also provide a comparison of the change in accuracies using 1-NN and SVM classifiers after applying ReliefF-Sc and Simba-Sc with ACS and RCG vs. varying the number of provided constraints.

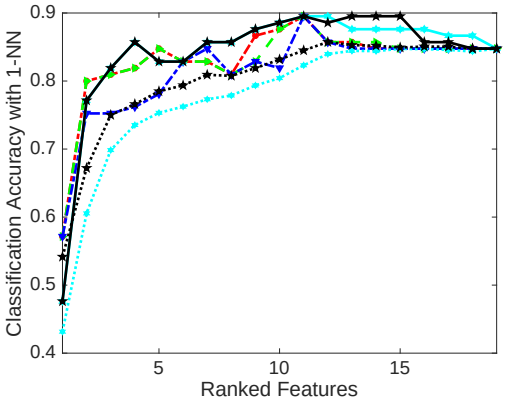
From Figure (4.8), we can see that generally both the constrained algorithms ReliefF-Sc and Simba-Sc were significantly improved when applied with ACS. We can also see that our ReliefF-Sc was not only improved with ACS to outperform its self when applied with RCG, but also to outperform Simba-Sc with ACS on the first few ranked features as can be seen in Figures (4.8a) and (4.8b) although both methods reached the same highest accuracy rate and on the same dimension on Wine dataset using 1-NN classifier. ReliefF-Sc with ACS was also able to compete with the supervised ReliefF and Fisher scores as can be seen in Figures (4.8c), (4.8d) and (4.8f). Note that, Fisher and ReliefF utilize full class labels and being able to compete with their performances only with few selected constraints is very interesting. In addition, from Figures (4.8c) and (4.8d) we can see that ReliefF-Sc and Simba-Sc with ACS both obtained prominent results on Segmentation, where they approximately maintained having higher accuracy curves over the whole set of ranked features compared to the supervised algo-



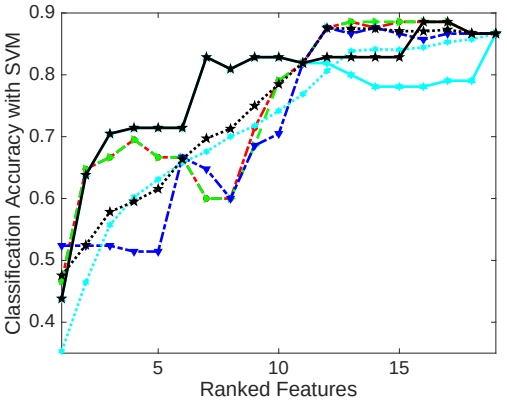
(a) Wine with 1NN classifier



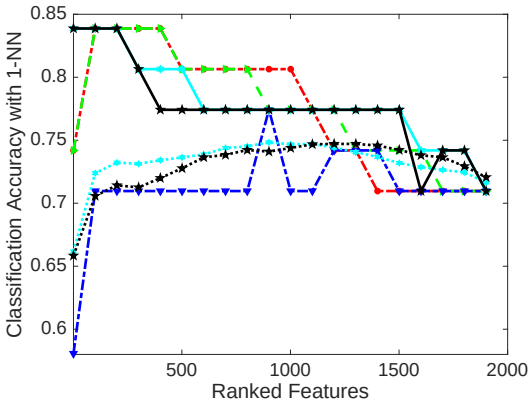
(b) Wine with SVM classifier



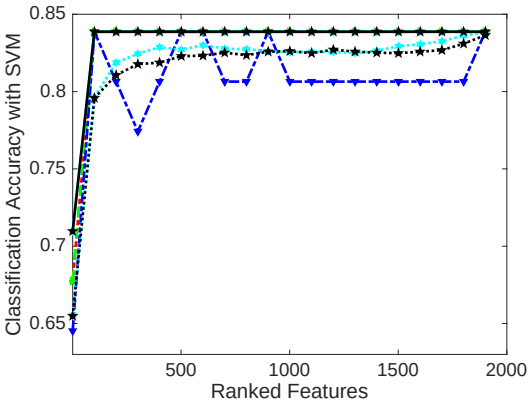
(c) Segmentation with 1NN classifier



(d) Segmentation with SVM classifier



(e) ColonCancer with 1NN classifier



(f) ColonCancer with SVM classifier

Figure 4.8: Accuracy rates using 1-NN and SVM classifiers vs. different number of selected features obtained by supervised (Fisher, ReliefF), unsupervised (Laplacian) and constrained ReliefF-Sc and Simba-Sc algorithms with RCG and ACS on 2 UCI datasets: (a,b) Wine; (c,d) Segmentation; and 1 high-dimensional gene-expression dataset: (e,f) ColonCancer.

rithms. It is also important to note that the accuracy curves obtained by ReliefF-Sc and Simba-Sc with RCG were sometimes below the accuracy curve obtained by the unsupervised Laplacian like in Figures (4.8a) and (4.8c), and this was completely resolved and enhanced for ReliefF-Sc when applied with ACS. Finally, from Figure (4.8e) on ColonCancer, we can see that the performance of ReliefF-Sc with RCG was improved significantly upon being applied with ACS to reach its highest possible accuracy rate from the first few features. Moreover, both ReliefF-Sc and Simba-Sc were able to obtain the same accuracy curves by the supervised Fisher and ReliefF with only 8 chosen constraints as can be seen in Figure (4.8f).

For clarity and to further compare all the constrained feature selection algorithms mentioned in section 4.6.1, we observe Table 4.4. First of all, from this table, we can see that feature selection by ReliefF-Sc, Simba-Sc, CS1, CS2, CS4, and CLS was generally always enhanced to obtain better highest accuracy rates on smaller selected dimensions when using ACS (rows shaded in gray). Also, we can see that ReliefF-Sc obtained the highest accuracy rate and on the smallest dimension compared to all other constrained algorithms in 4 experiments out of 6. It had a tie with Simba-Sc+ACS alone on Wine with 1-NN classifier and a tie with Simba-Sc, CS1, CS2, and CS4 with ACS on ColonCancer with 1-NN. However, on Wine and ColonCancer with SVM, ReliefF-Sc was the only algorithm that obtained the highest accuracy rate with the smallest dimension among all constrained algorithms. Moreover, although ReliefF-Sc with ACS did not win on Segmentation with both 1-NN and SVM, this loss was just in terms of the dimension and not the accuracy rate.

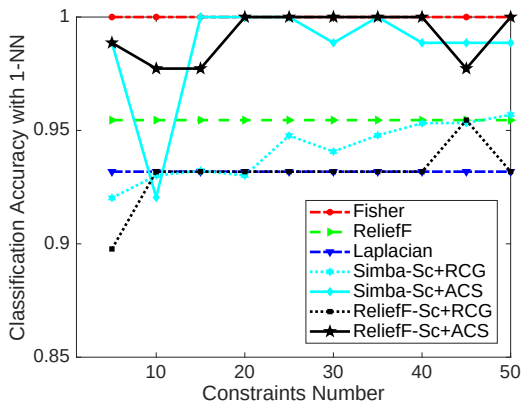
On another hand, since for margin-based algorithms, we are more interested in cannot-link constraints, we compare the performances obtained by ReliefF-Sc and Simba-Sc with ACS and RCG on Wine, Segmentation, and ColonCancer upon changing the number of constraints given to the algorithms. Figure (4.9) plots these classification accuracies taking the first 50% of the ranked features on each dataset. We present Fisher, ReliefF and Laplacian algorithms only as baselines since they do not use constraints. This figure generally shows stability in the accuracy obtained by ReliefF-Sc with ACS with respect to increasing the number of selected constraints, this shows the consistency in our constraint selection process. On the other side, it shows that Simba-Sc with ACS was not able to obtain such stability when the number of constraints is changed as can be seen in Figure (4.9b), this is due to its gradient ascent strategy that can obtain multiple solutions even for the same set of constraints. This means different runs might obtain different solutions even on the same input,

Table 4.4: The highest accuracy rates *Acc* (in %) and their corresponding dimensions *d* using 1-NN and SVM classifiers vs. different numbers of selected features obtained by supervised (Fisher, ReliefF), unsupervised (Laplacian) and constrained ReliefF-Sc, Simba-Sc, CS1, CS2, CS4 and CLS algorithms with RCG and ACS on 2 UCI datasets: Wine; Segmentation; and 1 high-dimensional gene-expression dataset: ColonCancer.

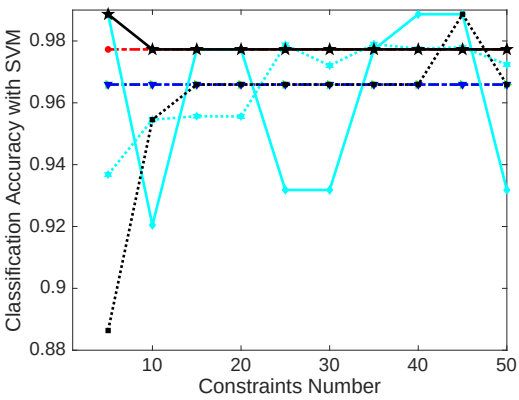
		1-NN Classifier						*SVM Classifier*					
		Wine		*Segmentation*		*ColonCancer*		*Wine*		*Segmentation*		*ColonCancer*	
		Acc	*d*	*Acc*	*d*	*Acc*	*d*	*Acc*	*d*	*Acc*	*d*	*Acc*	*d*
Without Selection		96.59	13	84.76	19	70.97	2000	97.73	13	86.67	19	83.87	2000
Fisher		100	5	89.52	11	83.87	5	97.72	5	88.57	13	83.87	11
ReliefF		98.86	10	89.52	11	83.87	6	97.72	3	88.57	13	83.87	20
Laplacian		100	6	89.52	11	77.42	8	97.72	6	87.61	12	83.87	49
ReliefF-Sc	*RCG*	97.75	11	85.73	12	74.87	1316	96.72	9	87.54	**12**	**83.87**	1945
	ACS	**100**	**5**	**89.52**	11	**83.87**	**1**	**98.86**	**4**	**88.57**	16	**83.87**	**19**
Simba-Sc	*RCG*	97.47	11	84.76	19	74.9	884	96.59	13	86.67	19	**83.87**	1926
	ACS	**100**	**5**	**89.52**	11	**83.87**	**1**	97.72	5	86.67	19	**83.87**	27
CS1	*RCG*	97.34	11	85.54	14	79.97	141	96.5	13	87.04	17	**83.87**	1878
	ACS	98.86	**5**	**89.52**	11	**83.87**	**1**	98.86	5	**88.57**	15	**83.87**	42
CS2	*RCG*	97	12	86.72	15	78.87	221	96.89	12	86.87	18	**83.87**	1913
	ACS	98.86	10	**89.52**	**7**	**83.87**	10	97.72	12	**88.57**	16	**83.87**	66
CS4	*RCG*	97.66	11	85.91	11	78.32	144	96.69	11	87.29	17	**83.87**	1892
	ACS	98.86	6	**89.52**	11	**83.87**	**1**	97.72	**4**	**88.57**	15	**83.87**	55
CLS	*RCG*	97.82	10	85.1	14	78.81	182	96.65	7	87.38	15	**83.87**	1823
	ACS	98.86	**5**	**89.52**	10	**83.87**	27	97.72	5	**88.57**	13	**83.87**	112

unlike the closed-form solution that ReliefF-Sc provides. Also it is important to note that when using ReliefF-Sc or Simba-Sc with RCG some added random constraints decreased the performance as can be seen in Figures (4.9d), (4.9e) and (4.9f), this proves that random constraints might introduce ill effects. In addition, (4.9a) shows that ReliefF-Sc and Simba-Sc with RCG obtained fluctuating accuracies that were generally below the supervised Fisher and ReliefF. On the other side, Figures (4.9c) and (4.9d) show that for all the number of constraints, ReliefF-Sc with ACS obtains stable and higher accuracies than those of the supervised Fisher and ReliefF. In addition, Figures (4.9e) and (4.9f) on ColonCancer similarly show that ReliefF-Sc with ACS had an accuracy equal to the one obtained by Fisher on all constraint numbers. This experiment shows the sensitivity of the feature selection process to the input constraints. Using more constraints does not mean higher accuracy, on the contrary, the quality of the constraints is what matters.

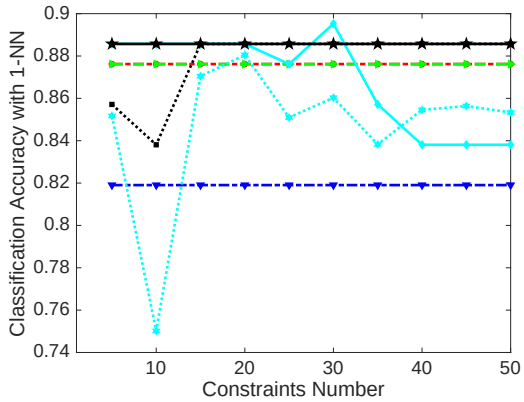
To sum up, Figure (4.8) and Table 4.4, highlight two very important findings. First, they clearly show that ACS is an algorithm-independent method since it was applied with multiple constrained feature selection algorithms and it was able to generally enhance and improve them. Second, they also prove that the combination of our ReliefF-Sc algorithm and our ACS method is able to outperform constrained and unsupervised feature selection methods and compete with the supervised ones. On the other side,



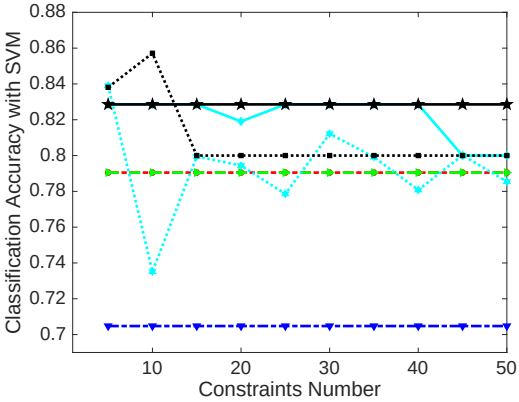
(a) Wine with 1NN classifier



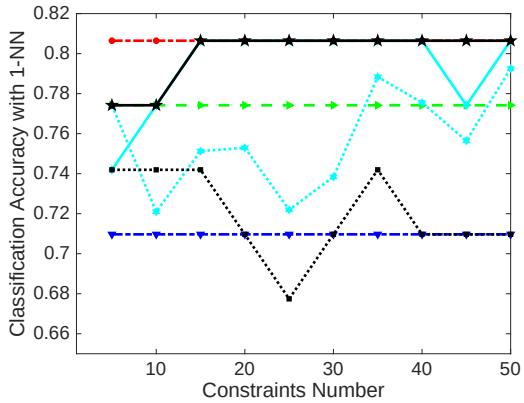
(b) Wine with SVM classifier



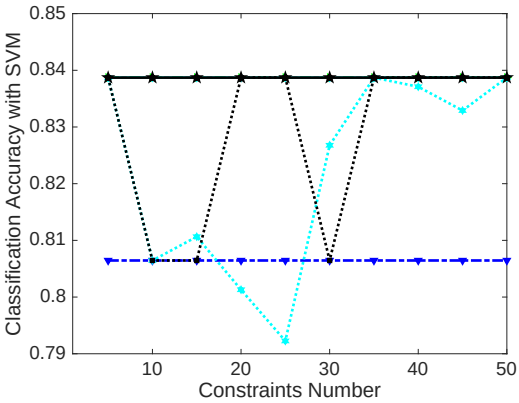
(c) Segmentation with 1NN classifier



(d) Segmentation with SVM classifier



(e) ColonCancer with 1NN classifier



(f) ColonCancer with SVM classifier

Figure 4.9: Accuracy rates obtained on the first 50% ranked features by the margin-based constrained ReliefF-Sc and Simba-Sc algorithms with RCG and ACS followed by 1-NN and SVM classifiers vs. different number of constraints on 2 UCI datasets: (a,b) Wine; (c,d) Segmentation; and 1 high-dimensional gene-expression dataset: (e,f) ColonCancer. Fisher, ReliefF and Laplacian do not use constraints and are used as supervised and unsupervised baselines.

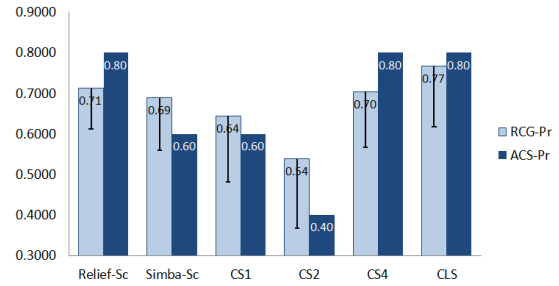
Figure (4.9) shows the importance of choosing the constraints systematically, proves the consistency provided by ACS, and points out the sensitivity of feature selection to the quality of inputted constraints.

In the following section, we further benchmark the same semi-supervised constrained algorithms with RCG and ACS but this time using the two feature-related performance measures (Precision and Distance).

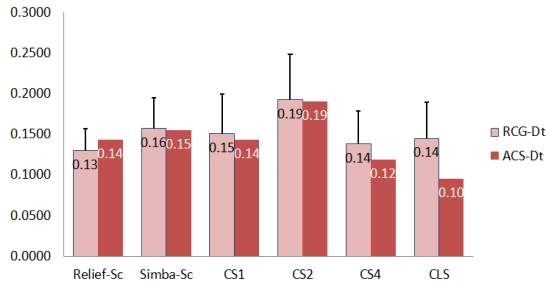
4.6.4.4 Precision and Distance measures by Different Constrained Feature Selection Methods with Random Generated Constraints (RCG) vs. with Actively Selected Constraints (ACS)

In this section, we compare the precision and distance measures obtained by the constrained feature selection algorithms. We use the ranked list of the supervised Fisher score as the reference (optimal) list in our comparisons as it generally obtained the highest accuracy rates in the above experiments and since it ranks features according to their correlation with the class labels directly. Thus, we consider it as the baseline by which we compare the ability of different semi-supervised algorithms to select the same features (precision) and in the same order (distance). Note that, we use the first half of the ranked lists by all algorithms in the calculation of these two measures assuming that this ensures the removal of irrelevant features (e.g. for ColonCancer we select first 1000 ranked features out of 2000 features in the original space). Also note that since for the RCG method the results are averaged over 100 repeated runs, we calculate the precision and distance measures for each run (on each subset we obtain) and we finally present the mean and standard deviations of these measures.

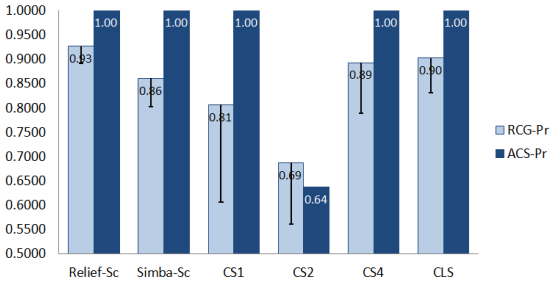
Figure (4.10) shows the bar graphs of precision and distance using the constrained algorithms with RCG and with ACS on Wine, Segmentation, and ColonCancer. It shows that generally, the ranked feature lists by constrained algorithms with active constraint selection obtained higher precision when compared to the supervised ranked feature list (by Fisher) with lower distances as can be seen in Figures (4.10e) and (4.10f) on ColonCancer. This means, even the order of selecting features tends to act in a supervised manner when constraints are selected actively. Also, the results on Segmentation in Figure (4.10c) show that constrained algorithms with ACS improved to identically act like a supervised feature selection algorithm with full class labels only with 10 selected constraints.
For instance, ReliefF-Sc, Simba-Sc, CS1, CS4, and CLS reached a precision value of 1 on this dataset. In such a case, when precision is maximum, even if the distance measure



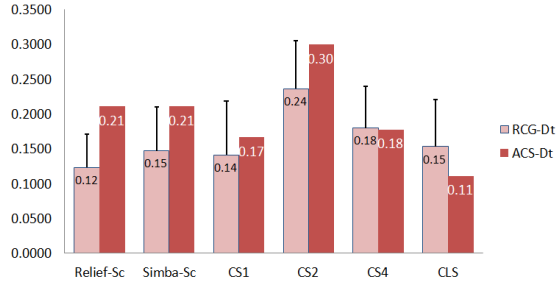
(a) Precision measure of feature subsets on Wine



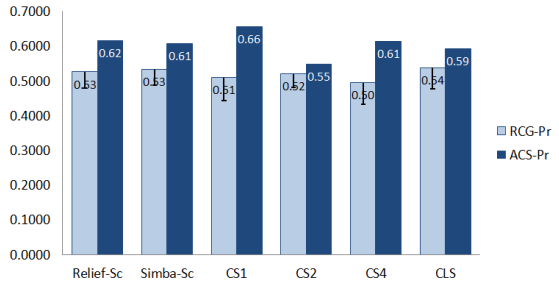
(b) Distance measure of feature subsets on Wine



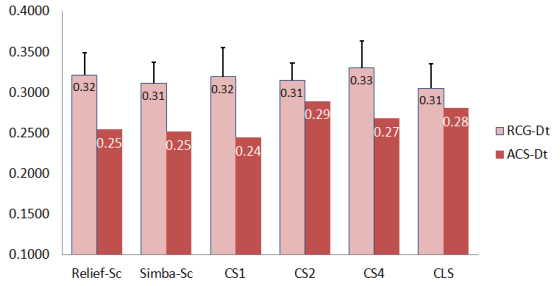
(c) Precision measure of feature subsets on Segmentation



(d) Distance measure of feature subsets on Segmentation



(e) Precision measure of feature subsets on Colon-cancer



(f) Distance measure of feature subsets on Colon-cancer

Figure 4.10: Three groups of two bar graphs each. Every two graphs in a row respectively show the Precision Pr and Distance Dt measures of the first 50% ranked features by ReliefF-Sc, Simba-Sc, CS1, CS2, CS4, and CLS with RCG and ACS on (a, b) Wine; (c, d) Segmentation ; (e, f) ColonCancer.

increases like for ReliefF-Sc and Simba-Sc with ACS in Figure (4.10d), this increase does not mean a performance degradation at all. It only means that the algorithms selected the optimal feature subset in another order. Although Simba-Sc, CS1, and CS2 showed a lower precision bar with ACS on Wine, it is important to notice that the standard deviation from the mean for RCG method is large enough to show that a lower precision is always possible and the high precision cannot be ensured. Similarly, in (4.10b) the standard deviation of the distance measure of RCG shows that it can increase to be higher than the fixed distance value obtained by our ACS method. On the other side, note that the precision obtained by ReliefF-Sc with ACS outperformed that of Simba-Sc, CS1, and CS2, whereas it was similar to that of CS4 and CLS on Wine. Also, all the constrained algorithms were competitive in terms of feature subsets' precision on Segmentation and ColonCancer except for CS2 that was lagging behind.

Finally, these experiments prove the significance of actively selecting constraints on the subset of selected features directly in addition to its importance and effect on the classification accuracy. As we have explained, Fisher is a supervised feature selection method that uses full class labels, thus, when ACS is able to enhance constrained feature selection algorithms to obtain similar feature subsets, it proves that only with few but well-chosen queries, constrained algorithms can obtain results that are able to compete with supervised ones.

4.7 Conclusion

In this chapter, we suggested a framework for actively selecting and propagating constraints for feature selection using graph Laplacian. In fact, we highlighted two main components: (1) our core contribution, i.e. the process of selecting pairwise constraints to be used by the constrained margin-based feature selection algorithm ReliefF-Sc and (2) the augmentation of supervision information by propagating these constraints. Focusing on the constraint selection process, we assumed that when a small perturbation in the similarity value of a data couple is able to perturb the similarity matrix and lead to a more well separated form of the second eigenvector of the graph Laplacian, this couple is definitely considered a more important and significant constraint. Thus, obtaining supervision information on such data couples is able to improve the performance of constrained feature selection algorithms. Moreover, for the sake of increasing supervision information without engaging the resource-expensive human oracle, we then propagated the selected constraints to their unlabeled neighborhood. This allowed the margin-based ReliefF-Sc to perform better with a sufficiently big set

of couples as it requires. Finally, experimental results showed how crucial constraint selection can be to the performance of constrained feature selection algorithms.

Conclusion and Perspectives

In this thesis, we provided a semi-supervised margin-based constrained feature selection framework that is capable of handling the main two aspects of feature selection: maximizing relevancy with respect to a target concept and minimizing redundancy. The latter is done for the sake of dimensionality reduction.

As we were interested in individual filter-type feature weighting/ranking, we provided a comprehensive review of a whole family of feature selection algorithms that use a margin-based score in order to evaluate and rank features in the order of their relevance. These are known as Relief-Based Algorithms (RBAs) and proved to generally perform well regardless of the problem specifications, they have a low bias, are considered relatively fast, are capable of detecting feature interaction, are robust and noise-tolerant in addition to being able to capture data local dependencies [1, 2]. These algorithms were initially suggested in the supervised context and extended to the semi-supervised one with class labels. However, feature selection became more challenging with the "small labeled-sample" problem, in which the amount of data that is unlabeled is much larger than the amount of labeled data taking into consideration the high cost of data labeling. Thus, we were interested in the semi-supervised constrained context where a cheaper kind of supervision information, i.e. pairwise constraints is used. These only specify whether two data points should be in the same group (must-link constraint), or in different groups (cannot- link constraint). So, we reviewed in details several recent works which have attempted to use pairwise constraints in both supervised and semi-supervised contexts.

With our interest in exploiting the strengths of margin-based feature selection within the semi-supervised constrained environment, we discussed margin-based feature selection with side pairwise constraints. We also demonstrated the change in the notion of margin from the supervised to the constrained context and provided its mathe-

matical interpretation. Accordingly, we suggested our constrained margin-based feature selection algorithm called Relief with Side constraints (Relief-Sc) and its robust version ReliefF-Sc. Focusing only on cannot-link constraints due to their conceptual importance from the margin's perspective, we showed how Relief-Sc utilizes these cannot-link constraints to assign weights to different features according to their contribution to maximizing the hypothesis-margin and finds a unique relevant feature subset in a closed-form. Experimental results proved that Relief-Sc is competitive to supervised scores using full class labels and can outperform other constrained and unsupervised ones. Moreover, we showed how it can provide good results with only a few pairwise constraints, which ensures computational advantages over other algorithms when dealing with high dimensional data.

Although the importance of constraints is practically proven, some constraints can degrade the learning performance. Thus, we suggested an Active Constraint Selection method (ACS) that depends on perturbation theory analysis to find a constraint set of higher utility. For that, we first discussed some of the most important related work on constraint selection. Then, we detailed the proposed method, by which, we find the data couple that is capable of reducing uncertainty in terms of spectral analysis. In other words, we find the couple that when supplied with background knowledge imposes the highest utility. Thus, we actively query the human oracle for information (pairwise constraint) on this particular chosen couple only.

Since we can only ask a small number of questions before the process becomes resource-expensive, we also presented a method for propagating the actively selected constraints into their unlabeled neighborhood, called PACS. In this regard, we showed how PACS can increase supervision information and enhance feature selection without requiring higher costs of human-labor.

On the other side, we suggested a novel combination of feature clustering upon a sparse graph with our margin-based objective function called Feature Clustering Relief-Sc (FCRSC). The efficiency of this combination was validated in comparison to supervised, unsupervised and semi-supervised filter feature selection methods using four different classifications schemes. The experimental results showed that eliminating redundant features can considerably improve the learning process.

Perspectives

Based on the outcomes presented in this thesis, we highlight several future research directions.

- In our work, Relief-Sc utilizes cannot-link pairwise constraints only. Thus, it might ignore valuable information carried by must-link constraints. Therefore, we can improve our work through finding a way to integrate must-link constraints in our constrained margin-based objective function.

- In ACS, we have focused on finding a constraint set of high utility with respect to our objective function. However, when a considerable number of points are near zero on the second eigen-vector (v_2), the process of selecting constraints can tend to be random. Therefore, we can mitigate this weakness by considering the sensitivity of the couples with respect to multiple eigenvectors instead of only the second eigenvector. Then, the couple of highest sensitivity on multiple eigenvectors is chosen. The number of the eigenvectors to be considered can be equal to the number of classes if known. In case not, we can either use all eigenvectors as can be seen in Equation (4.7) or use some method to automatically estimate the number of classes like eigengap [32] and density peaks [160].

- In ACS, we have focused on the First-Order Eigen-vector Perturbation Theory to choose the most significant couple. There is also some work that chooses features based on the eigenvalue sensitivity analysis [71]. It would be interesting to experiment whether eigenvalue sensitivity analysis could provide better performance in selecting our constraints.

- Our framework includes some computationally expensive subroutines such as calculating the sparse graph of the feature space when the number of features is very large and calculating eigenvector sensitivity when the number of data points increases significantly. Therefore, working on the scalability of our framework can be of great importance. For example, parallelization of the process of calculating eigenvector sensitivity matrix can be considered. The process can be naturally split into several independent tasks.

- Relief-Sc is a semi-supervised algorithm. Although it exploits unlabeled data in the neighborhood of a constraint, it is still closely related to the chosen constraint set. To decrease this effect, ensemble learning can be used where some other simple unsupervised algorithm that focuses on the intrinsic structure of data can be also used along with Relief-Sc. Each algorithm in the ensemble ranks the

features alone, at the end, some voting system finds the best in the combined results. Which means, the features ranked on top by both algorithms are the ones that will be considered.

- Application to Image analysis domain e.g. face recognition, texture.

Bibliography

[1] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. Recent advances and emerging challenges of feature selection in the context of big data. *Knowledge-Based Systems*, 86:33–45, September 2015.

[2] Ryan J. Urbanowicz, Melissa Meeker, William LaCava, Randal S. Olson, and Jason H. Moore. Relief-based feature selection: introduction and review. *Journal of Biomedical Informatics*, 85:189–203, September 2018.

[3] Dheeru Dua and Casey Graff. UCI Machine Learning Repository. School of Information and Computer Sciences, University of California, Irvine, USA, 2019. http://archive.ics.uci.edu/ml.

[4] Igor Kononenko. Estimating attributes: analysis and extensions of RELIEF. In *Proceedings of the European Conference on Machine Learning (ECML'94)*, pages 171–182, Catania, Italy, 6-8 April 1994.

[5] Marko Robnik-Šikonja and Igor Kononenko. Comprehensible interpretation of relief's estimates. In *Proceedings of the 18th International Conference on Machine Learning (ICML'01)*, pages 433–40, Williamstown, Massachusetts, USA, 28 June - 1 July 2001.

[6] Yijun Sun and Jian Li. Iterative RELIEF for feature weighting. In *Proceedings of the 23rd International Conference on Machine Learning (ICML'06)*, pages 913–920, Pittsburgh, Pennsylvania, USA, 25-29 June 2006.

[7] Trang T. Le, Ryan J. Urbanowicz, Jason H. Moore, and Brett A McKinney. STatistical Inference Relief (STIR) feature selection. *Bioinformatics*, 35(8):1358–1365, April 2019.

[8] Huan Liu and Hiroshi Motoda. *Computational methods of feature selection*. Data Mining and Knowledge Discovery Series. Chapman & Hall/CRC, Boca Raton, Florida, USA, October 2007.

[9] Jiliang Tang, Salem Alelyani, and Huan Liu. Feature selection for classification: A review. In *Data Classification: Algorithms and Applications*, chapter 2, pages 37–64. Chapman & Hall/CRC, Boca Raton, Florida, USA, January 2014.

[10] Razieh Sheikhpour, Mehdi Agha Sarram, Sajjad Gharaghani, and Mohammad Ali Zare Chahooki. A survey on semi-supervised feature selection methods. *Pattern Recognition*, 64:141–158, April 2017.

[11] Manoranjan Dash and Huan Liu. Feature selection for classification. *Intelligent Data Analysis*, 1(1-4):131–156, March 1997.

[12] Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):491–502, April 2005.

[13] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):94, December 2017.

[14] Jianyu Miao and Lingfeng Niu. A survey on feature selection. *Procedia Computer Science*, 91:919–926, December 2016.

[15] Benyamin Ghojogh, Maria N. Samad, Sayema Asif Mashhadi, Tania Kapoor, Wahab Ali, Fakhri Karray, and Mark Crowley. Feature selection and feature extraction in pattern analysis: A literature review. *Computing Research Repository (CoRR)*, abs/1905.02845, May 2019.

[16] Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, September 1936.

[17] Sebastian Mika, Gunnar Ratsch, Jason Weston, Bernhard Scholkopf, and Klaus-Robert Mullers. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop (Cat. No.98TH8468)*, pages 41–48, Madison, Wisconsin, USA, 25 August 1999.

[18] Elnaz Barshan, Ali Ghodsi, Zohreh Azimifar, and Mansoor Zolghadri Jahromi. Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds. *Pattern Recognition*, 44(7):1357–1371, July 2011.

[19] Karl Pearson. LIII. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, June 1901.

[20] Michael A.A. Cox and Trevor F. Cox. *Multidimensional scaling*. Chapman & Hall/CRC, Boca Raton, Florida, USA, 2008.

[21] Joshua B. Tenenbaum, Vin De Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[22] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 2000.

[23] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research (JMLR)*, 9:2579–2605, November 2008.

[24] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research (JMLR)*, 3:1157–1182, March 2003.

[25] Jie Cai, Jiawei Luo, Shulin Wang, and Sheng Yang. Feature selection in machine learning: A new perspective. *Neurocomputing*, 300:70–79, July 2018.

[26] Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th International Conference on Machine Learning (ICML'03)*, pages 856–863, Washington, District of Columbia, USA, 21-24 August 2003.

[27] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, June 2005.

[28] Yun Li, Tao Li, and Huan Liu. Recent advances in feature selection and its applications. *Knowledge and Information Systems*, 53(3):551–577, December 2017.

[29] Jiye Liang, Feng Wang, Chuangyin Dang, and Yuhua Qian. A group incremental approach to feature selection applying rough set technique. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):294–308, February 2014.

[30] Parham Moradi and Mehrdad Rostami. A graph theoretic approach for unsupervised feature selection. *Engineering Applications of Artificial Intelligence*, 44:33–45, September 2015.

[31] Mingxia Liu and Daoqiang Zhang. Sparsity score: A novel graph-preserving feature selection method. *International Journal of Pattern Recognition and Artificial Intelligence*, 28(04):1450009, June 2014.

[32] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007.

[33] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Qiang Yang, and Stephen Lin. Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 29(1):40–51, January 2007.

[34] Corinna Cortes and Mehryar Mohri. On transductive regression. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, pages 305–312, Vancouver, British Columbia, Canada, 4-7 December 2006.

[35] Zheng Zhao and Huan Liu. Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the 24th International Conference on Machine Learning*, pages 1151–1157, Corvalis, Oregon, USA, 20-24 June 2007.

[36] Fan R.K. Chung and Fan Chung Graham. *Spectral graph theory*, volume 92. American Mathematical Society, USA, 1997.

[37] Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning*, 53(1-2):23–69, October 2003.

[38] Xiaofei He, Deng Cai, and Partha Niyogi. Laplacian score for feature selection. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, volume 18, page 507–514, Vancouver, British Columbia, Canada, 5-8 December 2005.

[39] Zheng Zhao and Huan Liu. Semi-supervised feature selection via spectral analysis. In *Proceedings of the 7th SIAM International Conference on Data Mining*, pages 641–646, Minneapolis, Minnesota, USA, 26-28 April 2007.

[40] Daoqiang Zhang, Songcan Chen, and Zhi-Hua Zhou. Constraint score: A new filter method for feature selection with pairwise constraints. *Pattern Recognition*, 41(5):1440–1451, May 2008.

[41] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, pages 1601–1608, Vancouver, British Columbia, Canada, 13-18 December 2004.

[42] Zheng Alan Zhao and Huan Liu. *Spectral Feature Selection for Data Mining (Open Access)*. Data Mining and Knowledge Discovery Series. Chapman & Hall/CRC, Boca Raton, Florida, USA, December 2011.

[43] José Martínez Sotoca and Filiberto Pla. Supervised feature selection by clustering using conditional mutual information-based distances. *Pattern Recognition*, 43(6):2068–2081, June 2010.

[44] Saúl Solorio-Fernández, J. Ariel Carrasco-Ochoa, and José Fco Martínez-Trinidad. A review of unsupervised feature selection methods. *Artificial Intelligence Review*, pages 1–42, January 2019.

[45] Ahmed K. Farahat, Ali Ghodsi, and Mohamed S. Kamel. An efficient greedy method for unsupervised feature selection. In *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM)*, pages 161–170, Vancouver, British Columbia, Canada, 11-14 December 2011.

[46] Yun Li, Bao-Liang Lu, and Zhong-Fu Wu. A hybrid method of unsupervised feature selection based on ranking. In *Proceedings of the 18th IEEE International Conference on Pattern Recognition (ICPR'06)*, volume 2, pages 687–690, Hong Kong, China, 20-24 August 2006.

[47] Mingxia Liu, Dan Sun, and Daoqiang Zhang. Sparsity score: A new filter feature selection method based on graph. In *Proceedings of the 21st IEEE International Conference on Pattern Recognition*, pages 959–962, Tsukuba, Japan, 11-15 November 2012.

[48] V. Madhusudan Rao and V.N. Sastry. Unsupervised feature ranking based on representation entropy. In *Proceedings of the 1st IEEE International Conference on Recent Advances in Information Technology (RAIT)*, pages 421–425, Dhanbad, India, 15-17 March 2012.

[49] Kenji Kira and Larry A Rendell. The feature selection problem: Traditional methods and a new algorithm. In *AAAI*, pages 129–134, San Jose, California, USA, 12-16 July 1992.

[50] Yijun Sun. Iterative relief for feature weighting: algorithms, theories, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1035–1051, June 2007.

[51] Qin Liu, Jingxiao Zhang, Jiakai Xiao, Hongming Zhu, and Qinpei Zhao. A supervised feature selection algorithm through minimum spanning tree clustering. In *Proceedings of the 26th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 264–271, Limassol, Cyprus, 10-12 November 2014.

[52] Vinh Truong Hoang. *Multi color space LBP-based feature selection for texture classification*. PhD thesis, Université du Littoral Côte d'Opale, 15 February 2018.

[53] Mariam Kalakech, Philippe Biela, Ludovic Macaire, and Denis Hamad. Constraint scores for semi-supervised feature selection: A comparative study. *Pattern Recognition Letters*, 32(5):656–665, April 2011.

[54] Khalid Benabdeslem and Mohammed . Efficient semi-supervised feature selection: constraint, relevance, and redundancy. *IEEE Transactions on Knowledge and Data Engineering*, 26(5):1131–1143, May 2014.

[55] Yintong Wang, Jiandong Wang, Hao Liao, and Haiyan Chen. An efficient semi-supervised representatives feature selection algorithm based on information theory. *Pattern Recognition*, 61:511–523, January 2017.

[56] Xu-Kui Yang, Liang He, Dan Qu, and Wei-Qiang Zhang. Semi-supervised minimum redundancy maximum relevance feature selection for audio classification. *Multimedia Tools and Applications*, 77(1):713–739, January 2018.

[57] Ian Davidson, Kiri L. Wagstaff, and Sugato Basu. Measuring constraint-set utility for partitional clustering algorithms. In *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, pages 115–126, Berlin, Germany, 18-22 September 2006.

[58] Mohammed Hindawi, Kais Allab, and Khalid Benabdeslem. Constraint selection-based semi-supervised feature selection. In *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM)*, pages 1080–1085, Vancouver, British Columbia, Canada, 11-14 December 2011.

[59] Mohammed Hindawi. *Feature selection for semi-supervised data analysis in decisional information systems*. PhD thesis, INSA de Lyon, 21 February 2013.

[60] Md. Monirul Kabir, Md. Monirul Islam, and Kazuyuki Murase. A new wrapper feature selection approach using neural network. *Neurocomputing*, 73(16-18):3273–3283, October 2010.

[61] Ali El Akadi, Aouatif Amine, Abdeljalil El Ouardighi, and Driss Aboutajdine. A two-stage gene selection scheme utilizing MRMR filter and GA wrapper. *Knowledge and Information Systems*, 26(3):487–500, March 2011.

[62] Zhigang Ma, Yi Yang, Feiping Nie, Jasper Uijlings, and Nicu Sebe. Exploiting the entire feature space with sparsity for automatic image annotation. In *Proceedings of the 19th ACM International Conference on Multimedia*, pages 283–292, Scottsdale, Arizona, USA, 28 November - 1 December 2011.

[63] Caijuan Shi, Qiuqi Ruan, and Gaoyun An. Sparse feature selection based on graph laplacian for web image annotation. *Image and Vision Computing*, 32(3):189–201, March 2014.

[64] Christopher M. Bishop. *Neural Networks: A Pattern Recognition Perspective*. Oxford University Press and IOP Publishing, New York, USA, Handbook of Neural Computation edition, 1996.

[65] Manoranjan Dash and Huan Liu. Feature selection for clustering. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 110–121, Kyoto, Japan, 18-20 April 2000.

[66] Kenji Kira and Larry A. Rendell. A practical approach to feature selection. In *Proceedings of the 9th International Workshop on Machine Learning*, pages 249–256, Aberdeen, Scotland, 1-3 July 1992.

[67] Yi Yang, Heng Tao Shen, Zhigang Ma, Zi Huang, and Xiaofang Zhou. L2,1-norm regularized discriminative feature selection for unsupervised. In *Proceedngs of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*, Barcelona, Catalonia, Spain, 16-22 July 2011.

[68] Mark Andrew Hall. *Correlation-based feature selection for machine learning*. PhD thesis, University of Waikato, Hamilton, New Zealand, April 1999.

[69] Pablo A. Estévez, Michel Tesmer, Claudio A. Perez, and Jacek M. Zurada. Normalized mutual information feature selection. *IEEE Transactions on Neural Networks*, 20(2):189–201, February 2009.

[70] Huawen Liu, Jigui Sun, Lei Liu, and Huijie Zhang. Feature selection with dynamic mutual information. *Pattern Recognition*, 42(7):1330–1339, July 2009.

[71] Yi Jiang and Jiangtao Ren. Eigenvalue sensitive feature selection. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*, pages 89–96, Bellevue, Washington, USA, 28 June - 2 July 2011.

[72] Mahdokht Masaeli, Yan Yan, Ying Cui, Glenn Fung, and Jennifer G. Dy. Convex principal feature selection. In *Proceedings of the 10th SIAM International Conference on Data Mining (SDM)*, pages 619–628, Columbus, Ohio, USA, 29 April - 1 May 2010.

[73] Sebastián Maldonado and Richard Weber. A wrapper method for feature selection using support vector machines. *Information Sciences*, 179(13):2208–2217, June 2009.

[74] Saúl Solorio-Fernández, J. Ariel Carrasco-Ochoa, and José Fco Martínez-Trinidad. A new hybrid filter–wrapper feature selection method for clustering based on ranking. *Neurocomputing*, 214:866–880, November 2016.

[75] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, January 1967.

[76] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, Cambridge, UK, 2000.

[77] George H. John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pages 338–345, Montreal, Quebec, Canada, 18-20 August 1995.

[78] Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, California, USA, September 1993.

[79] Jidong Zhao, Ke Lu, and Xiaofei He. Locality sensitive semi-supervised feature selection. *Neurocomputing*, 71(10-12):1842–1849, June 2008.

[80] Chris Ding and Hanchuan Peng. Minimum redundancy feature selection from microarray gene expression data. *Journal of Bioinformatics and Computational Biology*, 3(02):185–205, May 2005.

[81] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, USA, December 1994.

[82] Zheng Zhao, Lei Wang, and Huan Liu. Efficient spectral feature selection with minimum redundancy. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 673–678, Atlanta, Georgia, USA, 11-15 July 2010.

[83] Jason Weston, André Elisseeff, Bernhard Schölkopf, and Mike Tipping. Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3:1439–1461, March 2003.

[84] Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, 5:1205–1224, December 2004.

[85] Zheng Zhao, Lei Wang, Huan Liu, and Jieping Ye. On similarity preserving feature selection. *IEEE Transactions on Knowledge and Data Engineering*, 25:619–632, October 2011.

[86] Inderjit S. Dhillon, Subramanyam Mallela, and Rahul Kumar. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Research*, 3:1265–1287, March 2003.

[87] Dino Ienco and Rosa Meo. Exploration and reduction of the feature space by hierarchical clustering. In *Proceedings of the 8th SIAM International Conference on Data Mining*, pages 577–587, Atlanta, Georgia, USA, 24-26 April 2008.

[88] Leo A. Goodman and William H. Kruskal. Measures of association for cross classifications III: Approximate sampling theory. *Journal of the American Statistical Association*, 58(302):310–364, June 1963.

[89] Wai-Ho Au, Keith C.C. Chan, Andrew K.C. Wong, and Yang Wang. Attribute clustering for grouping, selection, and classification of gene expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2(2):83–101, April 2005.

[90] Qinbao Song, Jingjie Ni, and Guangtao Wang. A fast clustering-based feature subset selection algorithm for high-dimensional data. *IEEE Transactions on Knowledge and Data Engineering*, 25(1):1–14, August 2011.

[91] Huawen Liu, Xindong Wu, and Shichao Zhang. Feature selection using hierarchical feature clustering. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM '11)*, pages 979–984, Glasgow, Scotland, UK, 24-28 October 2011.

[92] Xi Zhao, Wei Deng, and Yong Shi. Feature selection with attributes clustering by maximal information coefficient. *Procedia Computer Science*, 17:70–79, January 2013.

[93] Huan Liu and Hiroshi Motoda. *Feature extraction, construction and selection: A data mining perspective*. The Springer International Series in Engineering and Computer Science. Springer Science & Business Media, New York, USA, 1998.

[94] Yijun Sun, Sinisa Todorovic, and Steve Goodison. Local-learning-based feature selection for high-dimensional data analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1610–1626, September 2010.

[95] David W. Aha. Incremental constructive induction: An instance-based approach. In *Machine Learning Proceedings*, pages 117–121. Elsevier, Evanston, Illinois, USA, June 1991.

[96] James P. Callan, Tom Fawcett, and Edwina L. Rissland. CABOT: An adaptive approach to case-based search. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, pages 803–808, Sydney, New South Wales, Australia, 24-30 August 1991.

[97] Igor Kononenko, Edvard Šimec, and Marko Robnik-Šikonja. Overcoming the myopia of inductive learning algorithms with relieff. *Applied Intelligence*, 7(1):39–55, January 1997.

[98] Huan Liu and Hiroshi Motoda. Non-myopic feature quality evaluation with (R)relieff. In *Computational Methods of Feature Selection*, pages 169–191. Chapman and Hall/CRC, New York, USA, October 2007.

[99] Igor Kononenko, Marko Robnik-Sikonja, and Uros Pompe. ReliefF for estimation and discretization of attributes in classification, regression, and ILP problems. *Artificial Intelligence: Methodology, Systems, Applications*, pages 31–40, September 1996.

[100] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, December 1997.

[101] Marko Robnik-Šikonja and Igor Kononenko. An adaptation of Relief for attribute estimation in regression. In *Proceedings of the 14th International Conference on Machine Learning (ICML'97)*, pages 296–304, Nashville, Tennessee, USA, 8-12 July 1997.

[102] Ran Gilad-Bachrach, Amir Navot, and Naftali Tishby. Margin based feature selection-theory and algorithms. In *Proceedings of the 21st International Conference on Machine Learning*, pages 43–50, Banff, Alberta, Canada, 4-8 July 2004.

[103] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[104] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti A Zadeh. *Feature extraction: foundations and applications*. Springer, Berlin, Heidelberg, Germany, 2008.

[105] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, Pennsylvania, USA, 27-29 July 1992.

[106] Koby Crammer, Ran Gilad-Bachrach, Amir Navot, and Naftali Tishby. Margin analysis of the LVQ algorithm. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, pages 462–469, Vancouver, British Columbia, Canada, 9-14 December 2002.

[107] Ming Yang and Jing Song. A novel hypothesis-margin based approach for feature selection with side pairwise constraints. *Neurocomputing*, 73(16-18):2859–2872, October 2010.

[108] Bruce Draper, Carol Kaito, and José Bins. Iterative relief. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshop*, volume 6, pages 62–67, Madison, Wisconsin, USA, 16-22 June 2003.

[109] Abdelmalik Moujahid and Fadi Dornaika. Feature selection for spatially enhanced LBP: application to face recognition. *International Journal of Data Science and Analytics*, 5(1):11–18, February 2018.

[110] Christopher Stover and Eric W. Weisstein. Closed-form solution. `http://mathworld.wolfram.com/Closed-FormSolution.html`. Accessed: 2019-4-13.

[111] Edwin K.P. Chong and Stanislaw H. Zak. *An introduction to optimization*. Wiley series in discrete mathematics and optimization. John Wiley & Sons, Canada, fourth edition, January 2013.

[112] Geoffrey McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*. Wiley Series in Probability and Statistics. John Wiley & Sons, Canada, second edition, April 2007.

[113] Yubo Cheng, Yunpeng Cai, Yijun Sun, and Jian Li. Semi-supervised feature selection under Logistic I-Relief framework. In *Proceedings of the 19th International Conference on Pattern Recognition (ICPR)*, pages 1–4, Tampa, Florida, USA, 8-11 December 2008.

[114] Baige Tang and Li Zhang. Semi-supervised feature selection based on logistic I-RELIEF for multi-classification. In *Proceedings of the 15th Pacific Rim International Conference on Artificial Intelligence*, pages 719–731, Nanjing, China, 28-31 August 2018.

[115] Baige Tang and Li Zhang. Multi-class semi-supervised logistic I-RELIEF feature selection based on nearest neighbor. In *Proceedings of the 23rd Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 281–292, Macau, China, 14-17 April 2019.

[116] José Bins and Bruce A. Draper. Feature selection from huge feature sets. In *Proceedings of the 18th IEEE International Conference on Computer Vision (ICCV'01)*, pages 159–165, Vancouver, British Columbia, Canada, 7-14 July 2001.

[117] Yijun Sun, Sinisa Todorovic, and Steve Goodison. A feature selection algorithm capable of handling extremely large data dimensionality. In *Proceedings of the 8th SIAM International Conference on Data Mining*, pages 530–540, Atlanta, Georgia, USA, 24-26 April 2008.

[118] Samah Hijazi, Mariam Kalakech, Denis Hamad, and Ali Kalakech. Feature selection approach based on hypothesis-margin and pairwise constraints. In *Proceedings of the IEEE Middle East and North Africa Communications Conference (MENACOMM)*, pages 1–6, Jounieh, Lebanon, 18-20 April 2018.

[119] Catherine Krier, Damien François, Fabrice Rossi, and Michel Verleysen. Feature clustering and mutual information for the selection of variables in spectral data. In *Proceedings of the 15th European Symposium on Artificial Neural Networks (ESANN)*, pages 157–162, Bruges, Belgium, 25-27 April 2007.

[120] Jun Jiao, Xuan Mo, and Chen Shen. Image clustering via sparse representation. In *Proceedings of the 16th International Conference on Multimedia Modeling*, pages 761–766, Chongqing, China, 6-8 January 2010.

[121] Daniela M. Witten and Robert Tibshirani. A framework for feature selection in clustering. *Journal of the American Statistical Association*, 105(490):713–726, June 2010.

[122] Xiaojuan Huang, Li Zhang, Bangjun Wang, Zhao Zhang, and Fanzhang Li. Feature weight estimation based on dynamic representation and neighbor sparse reconstruction. *Pattern Recognition*, 81:388–403, September 2018.

[123] Yonghua Zhu, Xuejun Zhang, Ruili Wang, Wei Zheng, and Yingying Zhu. Self-representation and PCA embedding for unsupervised feature selection. *World Wide Web*, 21(6):1675–1688, November 2018.

[124] Lishan Qiao, Songcan Chen, and Xiaoyang Tan. Sparsity preserving projections with applications to face recognition. *Pattern Recognition*, 43(1):331–341, January 2010.

[125] Jin Xu, Guang Yang, Hong Man, and Haibo He. L1 graph based on sparse coding for feature selection. In *Proceedings of the 10th International Symposium on Neural Networks*, pages 594–601, Dalian, China, 4-6 July 2013.

[126] Xiaofeng Zhu, Xuelong Li, Shichao Zhang, Chunhua Ju, and Xindong Wu. Robust joint graph sparse coding for unsupervised spectral feature selection. *IEEE Transactions on Neural Networks and Learning Systems*, 28(6):1263–1275, June 2017.

[127] Chenping Hou, Feiping Nie, Xuelong Li, Dongyun Yi, and Yi Wu. Joint embedding learning and sparse regression: A framework for unsupervised feature selection. *IEEE Transactions on Cybernetics*, 44(6):793–804, June 2014.

[128] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Proceedings of the 14th International Conference on Neural Information Processing Systems*, pages 849–856, Vancouver, British Columbia, Canada, 3-8 December 2001.

[129] Jun Liu, Shuiwang Ji, and Jieping Ye. SLEP: Sparse learning with efficient projections. `https://github.com/divelab/slep/`, 2009.

[130] Tom Ronan, Zhijie Qi, and Kristen M Naegle. Avoiding common pitfalls when clustering biological data. *Science Signaling*, 9(432):re6–re6, June 2016.

[131] Uri Alon, Naama Barkai, Daniel A. Notterman, Kurt Gish, Suzanne Ybarra, Daniel Mack, and Arnold J Levine. Broad patterns of gene expression revealed

by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96(12):6745–6750, June 1999.

[132] Todd R. Golub, Donna K. Slonim, Pablo Tamayo, Christine Huard, Michelle Gaasenbeek, Jill P. Mesirov, Hilary Coller, Mignon L. Loh, James R. Downing, and Mark A. Caligiuri. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, October 1999.

[133] Se June Hong. Use of contextual information for feature ranking and discretization. *IEEE Transactions on knowledge and Data Engineering*, 9(5):718–730, October 1997.

[134] Huan Liu, Lei Yu, Manoranjan Dash, and Hiroshi Motoda. Active feature selection using classes. In *Proceedings of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 474–485, Seoul, Korea, 30 April - 2 May 2003.

[135] Arvind Kumar Shekar, Tom Bocklisch, Patricia Iglesias Sánchez, Christoph Nikolas Straehle, and Emmanuel Müller. Including multi-feature interactions and redundancy for feature ranking in mixed datasets. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 239–255, Skopje, Macedonia, 18-22 September 2017.

[136] Caiming Xiong, David M. Johnson, and Jason J. Corso. Active clustering with model-based uncertainty reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(1):5–17, January 2017.

[137] Zhengdong Lu and Miguel A. Carreira-Perpinan. Constrained spectral clustering through affinity propagation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Anchorage, Alaska, USA, 24-26 June 2008.

[138] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the 18th International Conference on Machine Learning (ICML'01)*, pages 577–584, Williamstown, Massachusetts, USA, 28 June - 1 July 2001.

[139] Pavan Kumar Mallapragada, Rong Jin, and Anil K. Jain. Active query selection for semi-supervised clustering. In *Proceedings of the 19th IEEE International Conference on Pattern Recognition (ICPR)*, pages 1–4, Tampa, Florida, USA, 8-11 December 2008.

[140] Ahmad Ali Abin and Hamid Beigy. Active selection of clustering constraints: a sequential approach. *Pattern Recognition*, 47(3):1443–1458, March 2014.

[141] Fabian L. Wauthier, Nebojsa Jojic, and Michael I. Jordan. Active spectral clustering via iterative uncertainty reduction. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1339–1347, Beijing, China, 12-16 August 2012.

[142] Khalid Benabdeslem, Haytham Elghazel, and Mohammed Hindawi. Ensemble constrained laplacian score for efficient and robust semi-supervised feature selection. *Knowledge and Information Systems*, 49(3):1161–1185, December 2016.

[143] Gilbert W. Stewart and J-G. Sun. *Matrix perturbation theory*. Academic Press., 1990.

[144] Dan Klein, Sepandar D. Kamvar, and Christopher D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. Technical report, Stanford, 2002.

[145] Qianjun Xu, Marie desJardins, and Kiri L. Wagstaff. Active constrained clustering by examining spectral eigenvectors. In *Proceedings of the 8th International Conference on Discovery Science*, pages 294–307, Singapore, 8-11 October 2005.

[146] Zhenyong Fu and Zhiwu Lu. Pairwise constraint propagation: A survey. *Computing Research Repository (CoRR)*, abs/1502.05752, February 2015.

[147] Sepandar D. Kamvar, Dan Klein, and Christopher D. Manning. Spectral learning. In *Proceedings of the 18th International Joint Conference of Artificial Intelligence*, page 561–566, Acapulco, Mexico, 9-15 August 2003.

[148] Zhiwu Lu and Horace HS Ip. Constrained spectral clustering via exhaustive and efficient constraint propagation. In *Proceedings of the 11th European Conference on Computer Vision*, pages 1–14, Crete, Greece, 5-11 September 2010.

[149] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, February 2007.

[150] Inmar Givoni and Brendan Frey. Semi-supervised affinity propagation with instance-level constraints. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pages 161–168, Clearwater Beach, Florida, USA, 16-18 April 2009.

[151] Zhenguo Li, Jianzhuang Liu, and Xiaoou Tang. Pairwise constraint propagation by semidefinite programming for semi-supervised classification. In *Proceedings of the 25th International Conference on Machine Learning*, pages 576–583, Helsinki, Finland, 5-9 July 2008.

[152] Olga Zoidi, Anastasios Tefas, Nikos Nikolaidis, and Ioannis Pitas. Person identity label propagation in stereo videos. *IEEE Transactions on Multimedia*, 16(5):1358–1368, August 2014.

[153] Denny Zhou, Olivier Bousquet, Thomas N. Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, pages 321–328, Vancouver, British Columbia, Canada, 8-13 December 2003.

[154] Xiang Wang, Jun Wang, Buyue Qian, Fei Wang, and Ian Davidson. Self-taught spectral clustering via constraint augmentation. In *Proceedings of the 14th SIAM International Conference on Data Mining*, pages 416–424, Philadelphia, Pennsylvania, USA, 24-26 April 2014.

[155] Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 4th SIAM International Conference on Data Mining*, pages 333–344, Lake Buena Vista, Florida, USA, 22-24 April 2004.

[156] Yi Jiang and Jiangtao Ren. Eigenvector sensitive feature selection for spectral clustering. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 114–129, Dublin, Ireland, 10-14 September 2011.

[157] Ling Huang, Donghui Yan, Nina Taft, and Michael I. Jordan. Spectral clustering with perturbed data. In *Proceedings of the 21st International Conference on Neural Information Processing Systems*, pages 705–712, Vancouver, British Columbia, Canada, 8-11 December 2008.

[158] Huazhong Ning, Wei Xu, Yun Chi, Yihong Gong, and Thomas S Huang. Incremental spectral clustering by efficiently updating the eigen-system. *Pattern Recognition*, 43(1):113–127, January 2010.

[159] Huan Liu, Hiroshi Motoda, and Lei Yu. A selective sampling approach to active feature selection. *Artificial Intelligence*, 159(1-2):49–74, November 2004.

[160] Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, June 2014.

[161] Suresh Balakrishnama and Aravind Ganapathiraju. Linear discriminant analysis-a brief tutorial. *Institute for Signal and information Processing*, 18:1–8, March 1998.

Publications

During this PhD thesis, two articles have been presented and published in international conferences. Also, two original articles were submitted to international journals.

International Conferences:

1. Hijazi, S., Kalakech, M., Hamad, D. and Kalakech, A. Feature selection approach based on hypothesis-margin and pairwise constraints. In IEEE Middle East and North Africa Communications Conference (MENACOMM), Jounieh, Lebanon, 18-20 April 2018, pp. 1-6.

2. Hijazi, S., Fawaz, R., Kalakech, M., Kalakech, A. and Hamad, D. Top development indicators for middle eastern countries. In IEEE Sixth International Conference on Digital Information, Networking, and Wireless Communications (DINWC), Beirut, Lebanon, 25-27 April 2018, pp. 98-102.

Accepted and submitted Articles to International Journal:

1. Hijazi, S., Hamad, D., Kalakech, M. and Kalakech, A. Active learning of constraints for weighted feature selection. Journal of Advances in Data Analysis and Classification. 40 pages. Accepted with revision, 2019.

2. Hijazi, S., Hamad, D., Kalakech, M. and Kalakech, A. A constrained feature selection approach based on feature clustering and hypothesis margin maximization. International Journal of Data Science and Analytics. 27 pages. Submitted on 12, January, 2020.

List of Tables

List of Figures

Abstract

Feature selection is a preprocessing step crucial to the performance of machine learning algorithms. It allows reducing computational costs, improving classification performances and building simple and understandable models. Recently, using pairwise constraints, a cheaper kind of supervision information that does not need to reveal the class labels of data points, received a great deal of interest in the domain of feature selection. Accordingly, we first proposed a semi-supervised margin-based constrained feature selection algorithm called Relief-Sc. It is a modification of the well-known Relief algorithm from its optimization perspective. It utilizes cannot-link constraints only to solve a simple convex problem in a closed-form giving a unique solution. However, we noticed that in the literature these pairwise constraints are generally provided passively and generated randomly over multiple algorithmic runs by which the results are averaged. This leads to the need for a large number of constraints that might be redundant, unnecessary, and under some circumstances even inimical to the algorithm's performance. It also masks the individual effect of each constraint set and introduces a human labor-cost burden.

Therefore, we suggested a framework for actively selecting and then propagating constraints for feature selection. For that, we made use of the similarity matrix based on Laplacian graph. We assumed that when a small perturbation of the similarity value between a data couple leads to a more well-separated cluster indicator based on the second eigenvector of the graph Laplacian, this couple is expected to be a pairwise query of higher and more significant impact. Constraints propagation, on the other side, ensures increasing supervision information while decreasing the cost of human labor.

Besides, for the sake of handling feature redundancy, we proposed extending Relief-Sc to a feature selection approach that combines feature clustering and hypothesis

margin maximization. This approach is able to deal with the two core aspects of feature selection i.e. maximizing relevancy while minimizing redundancy (maximizing diversity) among features.

Eventually, we experimentally validate our proposed algorithms in comparison to other known feature selection methods on multiple well-known UCI benchmark datasets which proved to be prominent. Only with little supervision information, the proposed algorithms proved to be comparable to supervised feature selection algorithms and were superior to the unsupervised ones.

Keywords: Dimensionality Reduction, Feature Selection, Cannot-link Constraints, Hypothesis-margin, Relief-Based Algorithm, Margin Maximization, Constraints selection, Active learning.

Résumé

Dans le domaine de l'apprentissage automatique, la sélection d'attributs est une étape d'une importance capitale. Elle permet de réduire les coûts de calcul, d'améliorer les performances de la classification et de créer des modèles simples et interprétables. Récemment, l'apprentissage par contraintes de comparaison, un type d'apprentissage semi-supervisé, a suscité un vif intérêt pour la sélection d'attributs. En effet, celui-ci est moins contraignant car il n'impose pas la connaissance des labels des classes. Dans ce contexte semi-supervisé avec contraintes, nous avons proposé un algorithme de sélection d'attributs à large marge appelé Relief-Sc. Il s'agit d'une modification de l'algorithme supervisé Relief. Il utilise uniquement les contraintes de comparaison cannot-links pour résoudre un problème d'optimisation convexe donnant une solution unique.

Les contraintes sont généralement générées aléatoirement, de manière passive et dans certains cas, défavorables aux performances de l'algorithme. Pour cela, nous proposons une méthodologie de sélection active des contraintes suivie d'une étape de propagation des contraintes. Nous avons appliqué la théorie de la perturbation sur la matrice de similarité du graphe Laplacien. Les contraintes cannot-links sont choisies parmi les couples de données ayant le plus d'influence sur la matrice de similarité. La procédure de propagation des contraintes est appliquée pour assurer une augmentation des informations de supervision tout en réduisant l'effort humain. De plus, dans un souci de gestion de la redondance des attributs, nous avons proposé d'étendre l'algorithme Relief-Sc en y intégrant une procédure de classification non supervisée des attributs. Cette approche permet de traiter les deux aspects fondamentaux de la sélection des attributs : maximiser la pertinence tout en minimisant la redondance (maximisation de la diversité) entre les attributs.

Finalement, nous avons validé expérimentalement les algorithmes proposés en les

comparant à d'autres algorithmes de sélection d'attributs sur plusieurs bases de données UCI. Nous avons montré qu'avec peu d'information de supervision, les performances des algorithmes proposés sont comparables aux algorithmes de sélection supervisée et supérieures aux algorithmes non supervisés.

Mots-clés: Réduction de la dimension, sélection d'attributs, contraintes de comparaison, algorithme Relief, marge maximale, apprentissage actif.

Résumé Étendu de la Thèse

La croissance rapide des technologies a entraîné une augmentation exponentielle de la quantité de données générées dans divers domaines. Par exemple, dans des domaines tels que les réseaux sociaux, la santé, le marketing, la bio-informatique et la biométrie, les données fournies telles que les images, les vidéos, les textes, la voix, les micros-réseaux d'expression des gènes et d'autres types obtenus à partir des relations sociales et de l'Internet des objets peuvent non seulement être énormes en termes de nombre d'instances, mais également en termes de dimensionnalité des attributs. Cela pose de nombreux défis pour une gestion efficace des données.

Par conséquent, l'utilisation des techniques d'exploration de données et d'apprentissage automatique est devenue une nécessité pour l'extraction automatique des connaissances et la découverte des modèles sous-jacents dans ces données. En fait, selon les connaissances contextuelles et la manière d'identifier les modèles de données, ces techniques peuvent être classées en trois catégories: la classification, la régression (ou prévision) et le regroupement (clustering en anglais). La classification des données, dont le but est d'identifier l'appartenance d'un point de données à un groupe prédéfini, peut modéliser de nombreuses applications dans le monde réel. En effet, les ensembles de données sont généralement représentés par des matrices bidimen-

sionnelles où les lignes correspondent aux échantillons de données et les colonnes correspondent aux attributs qui les caractérisent.

En introduisant des données de grandes dimensions dans le «modèle» des techniques d'apprentissage traditionnelles sans pré-traitement approprié, des performances d'apprentissage insatisfaisantes peuvent être obtenues en raison d'un problème critique connu sous le nom de malédiction de la dimensionnalité [8]. Ce dernier fait référence à un phénomène selon lequel les données deviennent plus dispersées dans les espaces de grandes dimensions, ce qui conduit à sur-adapter l'algorithme d'apprentissage [9]. De plus, certains attributs caractérisant les données n'apportent aucune information utile à un critère d'évaluation de la pertinence (par exemple, la discrimination de classe). D'autres attributs peuvent être corrélés ou redondants, ce qui rend le processus d'apprentissage complexe, coûteux en termes de stockage et de calcul, inefficace, moins généralisable et difficile à interpréter.

Les attributs d'entrée d'origine sont généralement composés des quatre groupes suivants: (a) complètement non-pertinents, (b) faiblement pertinents et redondants, (c) faiblement pertinents mais non redondants, et (d) attributs fortement pertinents [25]. Les deux premiers groupes peuvent dégrader considérablement les performances des algorithmes d'apprentissage et diminuer leur efficacité de calcul [28–30]. Un algorithme de sélection d'attributs convenable devrait permettre de conserver les attributs au sein des groupes (c) et (d). La figure (A) présente les quatre groupes existants et identifie le sous-ensemble d'attributs souhaité optimal.

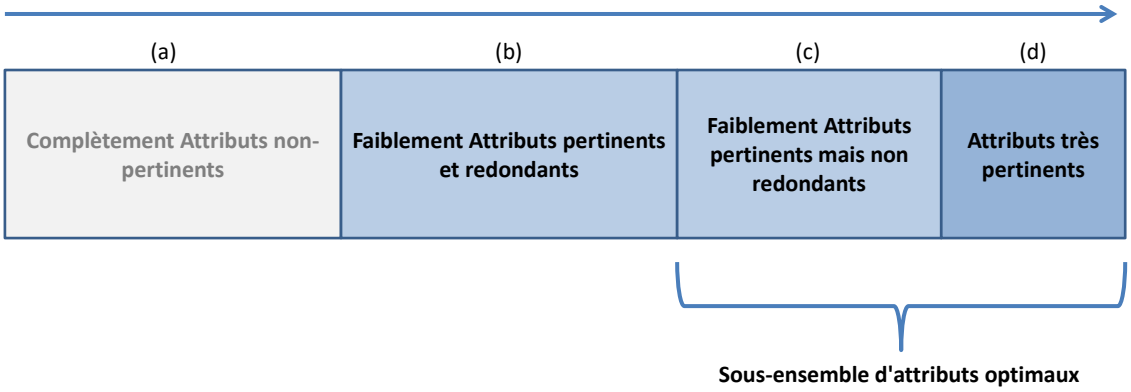


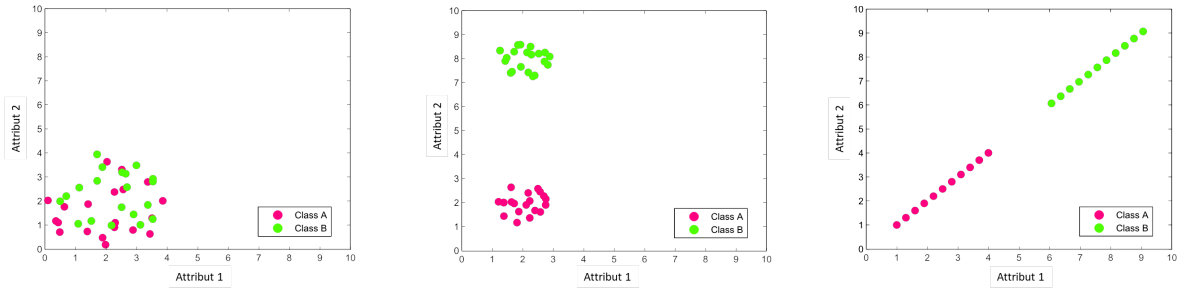
Figure A : Les quatre types d'attributs selon leur pertinence et leur redondance.

Pour mieux expliquer les notions de pertinence et de redondance, nous utilisons deux attributs pour présenter trois exemples différents illustrés dans la Figure (B). Le premier exemple présenté dans la Figure (B a) montre le cas où l'attribut 1 et l'attribut 2 sont considérés comme non-pertinents, à cause de leur incapacité à discriminer les points de données des deux classes. Dans le deuxième exemple, présenté dans la Figure (B b), l'attribut 1 est considéré comme non-pertinent; alors que l'attribut 2 peut clairement séparer la classe A de la classe B, et par conséquence, il est considéré comme pertinent. Généralement, l'attribut est dit pertinent quand il décrit mieux un ensemble de données en fonction d'un certain critère d'évaluation de pertinence spécifique, ce qui correspond à la capacité de séparation de classes (problèmes de classification) dans notre exemple. Par contre, le troisième exemple, présenté dans la Figure (B c), correspond au cas où l'attribut 1 et l'attribut 2 sont redondants. Dans ce cas, les informations fournies par les attributs sont fortement corrélées.

Il est bien connu que la présence d'attributs non pertinents et redondants peut

pénaliser les performances d'un algorithme d'apprentissage automatique, augmenter ses besoins en stockage, augmenter ses coûts de calcul et rendre la visualisation des données et l'interprétabilité des modèles plus difficiles. Pour atténuer ces problèmes, la réduction de dimensionnalité en tant que stratégie de prétraitement des données est l'un des outils les plus puissants à utiliser. Elle peut être principalement divisée en deux groupes différents, l'extraction d'attributs et la sélection d'attributs [10–12].

- **Extraction d'attributs**: projette l'espace d'entrée initial sur une nouvelle dimension inférieure en combinant les attributs d'origine de manière linéaire ou non linéaire, modifiant ainsi leur signification [13, 14]. En fait, l'extraction linéaire d'attributs est appliquée lorsque les données sont dans un sous-espace linéaire ou lorsque les classes de données peuvent être discriminées linéairement, alors que l'extraction non-linéaire d'attributs est appliquée lorsque le modèle de données est supposé être plus complexe et existe sur une sous-variété non-linéaire [15]. Certains algorithmes d'extraction d'attributs bien connus sont l'analyse dis-



(a) (b) (c)

Figure B : Trois exemples illustrant les notions de pertinence et de redondance. (a) montre deux attributs non-pertinents; (b) montre un attribut pertinent (attribut 2) et un attribut non-pertinent (attribut 1); et (c) montre deux attributs redondants.

criminante linéaire (LDA) [161], l'analyse en composantes principales (PCA) [19] et les algorithmes à base de graphes.

- **Sélection d'attributs**: la sélection d'attributs vise à sélectionner les attributs qui décrivent au mieux une base de données parmi un grand ensemble d'attributs candidats, afin de créer un espace de dimension inférieure, sans aucune transformation ni modification de la signification d'attributs d'origine [12, 14, 24, 25]. Pour les problèmes de classification, la sélection des attributs vise à sélectionner les attributs les plus discriminants. En d'autres termes, il s'agit de sélectionner les attributs qui distinguent mieux les points de données appartenant à différentes classes. Parmi les algorithmes de sélection d'attributs les bien connus, on peut citer le coefficient de corrélation de Pearson (PCC) [24], l'information mutuelle (MI) [15], le filtre à corrélation rapide (FCBF) [26], la sélection séquentielle en avant (SFS), la sélection séquentielle en arrière (SBS) [15] et la Redondance minimale Pertinence Maximale (mRMR) [27].

Les deux outils de réduction de dimensionnalité ci-dessus, l'extraction d'attributs et la sélection d'attributs, sont efficaces et capables de:

- Améliorer les performances d'apprentissage.
- Diminuer la mémoire de stockage.
- Améliorer l'efficacité des calculs.
- Construire de meilleurs modèles de généralisation.

Néanmoins, la sélection d'attributs coûte moins cher en terme de calcul et elle

est meilleure en termes de lisibilité et d'interprétabilité. Alors que l'extraction d'attributs projète l'espace d'attributs d'origine dans un nouvel espace de dimension inférieure. Contrairement à la sélection d'attributs, la signification des attributs extraits n'est pas conservée. Il est important de noter qu'il est parfois crucial de conserver le sens original des attributs, comme dans l'analyse génétique, qui consiste à déterminer quels gènes sont responsables d'une maladie donnée [8, 24]. Par conséquent, dans notre travail, nous nous intéressons aux méthodes de sélection d'attributs.

Généralement, la sélection individuelle d'attributs peut être réalisée en associant à chaque attribut un score de pertinence qui est un poids positif ou nul et en les ordonnant selon leurs scores. En effet, le processus d'évaluation de ce score change avec le changement de connaissance contextuelle. Par conséquent, dans un contexte non supervisé où aucune information de labellisation n'est disponible, les méthodes de sélection d'attributs ont recours à des approches de similarité des données et d'informations discriminantes locales pour mesurer la capacité des attributs dans des groupes de données discriminantes. D'un autre côté, dans un contexte supervisé où les données sont entièrement labellisées, les méthodes de sélection d'attributs accordent des scores plus élevés aux attributs ayant des mesures de corrélation élevées avec les labellisations de classes.

Il existe toute une famille d'algorithmes de sélection d'attributs qui utilisent un score basé sur la marge afin d'évaluer et de classer les attributs. Ceux-ci sont appelés algorithmes à base de Relief (RBAs) et ont été initialement suggérés dans le con-

texte supervisé pour des problèmes à deux classes. Ils affectent des poids plus importants aux attributs qui contribuent au mieux à élargir une distance interclasses appelée l'hypothèse de marge. Cette marge est calculée en tant que différence entre la distance d'un point à son "nearmiss" (point le plus proche ayant une labellisation différente) et la distance à son "nearhit" (point le plus proche ayant la même labellisation). Les RBAs se sont généralement bien comportés, quelles que soient les spécifications du problème. Ils ont des algorithmes de filtrage à faible biais (indépendants des classifieurs), considérés relativement rapides, capables de détecter l'interaction d'attributs, robustes et tolérants au bruit, en plus de leur capacité à réveler les dépendances locales des données [1, 2].

Toutefois, dans de nombreuses applications du monde réel, il existe une faible quantité de données labellisés et beaucoup plus de données non labellisés, et par conséquent, l'emploi unique d'algorithmes supervisés ou non supervisés de sélection d'attributs ne peuvent pas profiter de tous les points de données dans ces scénarios. Pour cela, Il était bénéfique d'utiliser des méthodes semi-supervisées pour exploiter les points étiquetés et non étiquetés. En fait, par rapport aux labellisations des classes, les contraintes par paires sont un autre type d'information de supervision qui peuvent être acquises plus facilement. Ces contraintes spécifient simplement si une paire de points de données appartient à la même classe (must-link constraint) ou à des classes différentes (cannot-link constraint), sans spécifier les classes elles-mêmes. De nombreux scores de contraintes ont utilisé ces deux notions pour classer les attributs, mais ils négligent souvent les informations fournies par des données non contraintes et non

labellisées.

Cela nous a amenés à proposer un nouveau cadre pour la sélection d'attributs semi-supervisée avec contraintes basées sur des marges, et qui gère les deux aspects essentiels de la sélection d'attributs: la pertinence et la redondance. Ce cadre est divisé en trois composants principaux.

Le premier composant consiste en un algorithme de sélection d'attributs semi-supervisés avec contraintes basées sur des marges qui utilisent uniquement des contraintes de comparaison de type cannot-link, et bénéficient à la fois du voisinage local non étiqueté des points de données ainsi que des contraintes fournies. Pour cela, nous avons suggéré d'intégrer la modification du score de sélection basé sur l'hypothèse de marges utilisé avec des contraintes cannot-link, avec la solution analytique de l'algorithme de Relief supervisé [50, 66]. Autrement dit, nous avons fusionné le concept de l'hypothèse de marges, utilisé dans [107] avec l'algorithme Relief pour obtenir une meilleure optimisation. Par conséquent, nous obtenons notre algorithme de sélection d'attributs semi-supervisé basé sur des marges, appelé **Relief-Sc**. Il utilise des contraintes cannot-link pour résoudre un problème convexe simple donnant une solution unique. En outre, nous présentons l'algorithme **Relief-Sc** légèrement modifié dans une version plus robuste, appelée algorithme **ReliefF-Sc**.

Pour expliquer l'idée de base de **Relief-Sc**, considérons $(\boldsymbol{x}_n, \boldsymbol{x}_n)$ un couple de contraintes de cannot-link avec $\boldsymbol{x}_n$ et $\boldsymbol{x}_m$ représentant deux points de données. **Relief-Sc**

trouve ensuite les points les plus proches (nearhit) de x_n et x_m. Ensuite, il calcule la différence entre la distance entre x_n et le nearhit de x_m et entre x_n et son propre nearhit. Notez que cette différence est observée sur chacun des attributs disponibles. Ainsi, **Relief-Sc** évalue la "marge" induite par chaque attribut. En d'autres termes, il suppose que la capacité d'un attribut à discriminer entre les points de données puisse être appliquée en fonction de sa contribution à la maximisation de la marge. Cette contribution peut être représentée et quantifiée par un vecteur de poids s'étendant sur tous les attributs [118]. Ayant un ensemble de contraintes de cannot-link, **Relief-Sc** met à jour de manière itérative le poids de chaque attribut en fonction de la marge qu'il induit sur chaque contrainte de cannot-link. À la fin, les attributs ayant le poids les plus élevés sont classés en premier et par conséquent considérés comme plus pertinents pour la fonction objective considérée.

Cependant, nous avons constaté que les contraintes par paires sont généralement dites passives et sont générées aléatoirement à partir des données labellisées pour évaluer la moyenne des résultats. Cela conduit à la génération aléatoire d'un grand nombre de contraintes qui peuvent être redondantes, inutiles, et même, dans certaines circonstances, défavorables aux performances de l'algorithme. Ce qui masque également l'effet individuel de chaque ensemble de contraintes.

Sélection active des contraintes

En outre, bien qu'il soit communément admis que l'ajout de contraintes en tant qu'information de supervision améliorerait les performances du clustering, Davidson et al.

[57] ont mentionné que les performances pourraient être dégradées, même lorsque les contraintes sont aléatoires, mais générées directement à partir de données labellisées. Par conséquent, nous nous attendions à ce que les attributs utilisés puissent affecter la performance de la méthode de sélection des attributs avec contraintes, de la même manière qu'ils affectent le clustering avec contraintes.

En conséquence, nous avons suggéré le deuxième composant de notre cadre dans lequel nous avons ciblé le processus de sélection systématique et active des contraintes sur la base de l'idée de perturbation de matrice de similarité [143]. Pour cela, nous bénéficions des caractéristiques spectrales du graphe Laplacien qui est à son tour défini sur la matrice de similarité. L'impact de chaque couple de données (contraintes de comparaison) sur cette matrice peut être reflété par le changement qu'il peut entraîner sur le graphe Laplacien, et en particulier sur ses valeurs et vecteurs propres. Lorsqu'une petite perturbation de la valeur de similarité d'un couple est capable de perturber le graphe Laplacien conduisant à une forme mieux séparée du second vecteur propre, ce couple est définitivement considéré comme une contrainte plus importante et plus significative. Nous avons donc proposé une méthode de sélection de contrainte active (**ACS**) basée sur un critère de sensibilité du second vecteur propre.

Techniquement, l'algorithme **ACS** commence initialement par une matrice complète de similarités entre les points de données à partir desquels il calcule le graphe Laplacien des données. Il trouve ensuite les vecteurs propres et les valeurs propres de ce dernier graphe suivis en considérant spécifiquement le deuxième vecteur pro-

pre. Lorsque les données sont organisées en groupes, leur projection sur le deuxième vecteur propre est suffisante pour diviser l'ensemble de données en deux groupes distincts. Ces deux groupes sont obtenus en seuillant les valeurs du deuxième vecteur propre par rapport à zéro. Ainsi, un groupe est formé sur le côté négatif et l'autre sur le côté positif montrant que ce vecteur propre joue le rôle d'une approximation du vecteur indicateur de clusters. **ACS** se concentre ensuite sur la capture du point de magnitude minimum sur le deuxième vecteur propre estimé. Ce point est considéré comme celui qui présente la plus grande incertitude. Il s'ensuit alors que **ACS** utilise de manière itérative le théorème de l'analyse de perturbations pour trouver activement le couple dans l'espace d'entrée qui a la plus grande dérivée de premier ordre par rapport à ce point minimum sur le deuxième vecteur propre (celui le plus proche de zéro est considéré le plus incertain) [156, 157].

Par conséquent, **ACS** mesure ensuite la sensibilité du point le plus incertain sur le deuxième vecteur propre par rapport au changement des valeurs de similarité entre chaque couple de données. Une fois qu'il trouve le couple de points le plus incertain sur le deuxième vecteur propre, seul ce couple est interrogé par comparaison et sa valeur de similarité correspondante est modifiée. Pour cela, une source externe de connaissances (par exemple un expert) est invitée à observer le couple choisi et à fournir une réponse sur la question de savoir si le couple se ressemble must-link ou ne se ressemble pas cannot-link. S'il apparaît que le couple interrogé doit être une contrainte de cannot-link (or must-link), sa valeur de similarité passe à 0 (ou 1). Enfin, le deuxième vecteur propre est à nouveau calculé et le processus est reitéré jusqu'à trouver le nombre de contraintes requis.

Alors, notre objectif était de rechercher les couples de données dont les projections sur le deuxième vecteur propre sont les plus proches de zéro et de signes opposés. Ces derniers couples seront les seuls interrogés sur les contraintes, réduisant ainsi le coût des requêtes. En fait, ces contraintes, qui peuvent réduire l'incertitude et améliorer la séparation des classes, permettent de sélectionner des attributs pertinents dotés de la plus grande capacité de discrimination de classes.

Propagation des contraintes

De plus, comme nous ne pouvons poser que quelques questions avant que le processus ne devienne ennuyeux et coûteux en ressources, nous avons également montré le processus de propagation des contraintes sélectionnées activement dans leur voisinage non-labellisé, appelé **PACS**. À cet égard, nous avons montré comment le **PACS** peut augmenter les informations de supervision et améliorer la sélection des attributs sans nécessiter de coûts de de requêtes plus élevés. Cela est réalisé en décomposant le problème en un ensemble de sous-problèmes de propagation de labellisations indépendantes.

Dans le **PACS**, les contraintes de cannot-link initialement obtenues par **ACS** sont représentées par une matrice d'indicateurs parcimonieux qui a une valeur de 1 uniquement entre les couples de cannot-link. Ensuite, la propagation des contraintes verticales, c.-à-d., le long des colonnes de la matrice de contraintes initiale, est appliquée. Parfois, une colonne peut ne contenir aucune contrainte par paire, cela signifie que

toutes les cellules de cette colonne peuvent avoir la valeur zéro et donc aucune propagation de contrainte verticale ne peut être obtenue pour cette colonne particulière. Ceci est résolu par une étape de propagation horizontale. Cette dernière est appliquée après avoir terminé la propagation verticale dans une approche itérative et conduit à une matrice complète qui contient une contrainte douce entre tous les deux points de données. Finalement, il est nécessaire de trouver un seuil approprié pour filtrer les contraintes propagées. Ainsi, tout couple correspondant à un élément de la matrice de contraintes souples ayant une valeur inférieure au seuil sera ignoré et tout couple correspondant à une élément ayant une valeur supérieure ou égale au seuil sera considéré comme une contrainte de cannot-link dans le nouvel ensemble de contraintes propagées. Ce seuil est calculé comme la moyenne des valeurs maximales obtenues à partir de chaque ligne de la matrice complète des contraintes souples.

De plus, nous avons remarqué qu'un inconvénient de l'algorithme **Relief-Sc**, hérité de sa version standard supervisée [66], est qu'il n'est pas assez efficace pour gérer la redondance entre les attributs. Néanmoins, il est bien connu que l'élimination des attributs redondants est également un aspect important de la sélection des attributs.

Sélection d'attributs représentatifs

En conséquence, nous avons suggéré le troisième composant de notre cadre dans lequel on étend notre méthode de sélection d'attributs semi-supervisée à une nouvelle combinaison de regroupement d'attributs et de maximisation des marges de l'hypothèse appelée **FCRSC**. Cette approche, présentée dans la figure (C), vise à traiter

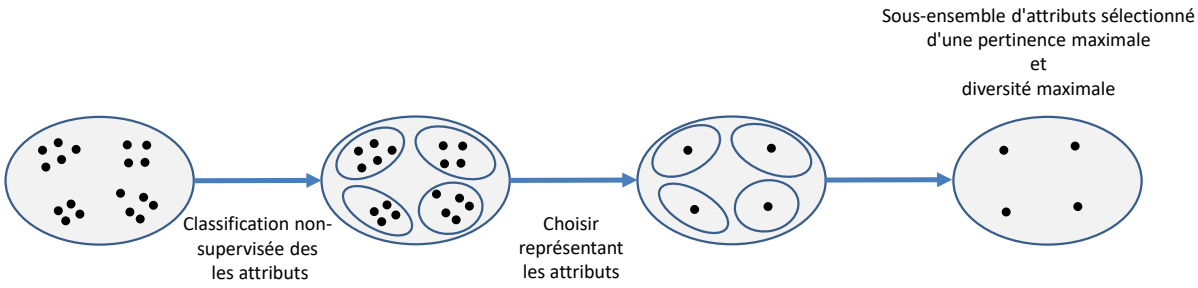


Figure C : Schéma méthodologique de sélection d'attributs représentatifs de clusters dans l'espace d'attributs.

les deux aspects essentiels de la sélection des attributs (pertinence et redondance) et est divisée en trois étapes.

- Étape 1: Une mesure de distance appropriée est choisie pour évaluer la similarité ou la dissimilarité entre les attributs. Dans le **FCRSC** les poids de similarité entre les attributs sont représentés par un graphe parcimonieux en vertu duquel on suppose que chaque attribut peut être reconstruit à partir de la combinaison linéaire parcimonieuse d'autres attributs. Cela se fait en résolvant un L_1-problème de minimisation.

- Étape 2: Une méthode de clustering appropriée est appliquée pour regrouper des attributs similaires. Parmi les quatre principales catégories de regroupement qui sont hiérarchiques, basées densité, statistiques et basées centroïdes [130], nous étions intéressés par le regroupement hiérarchique. Cet algorithme génère un diagramme arborescent appelé dendrogramme. Intuitivement, à des niveaux inférieurs du dendrogramme, nous avons les grappes des attributs les plus redondants qui ont d'abord été regroupés. Ainsi, la réduction à de faibles niveaux

entraîne un nombre plus élevé de clusters et donc davantage de représentants de clusters, c.-à-d., un sous-ensemble d'attributs de sortie plus important. Par conséquent, nous avons décidé de couper l'arbre lorsque les distances de fusion deviennent suffisamment grandes pour créer une hiérarchie de deuxième niveau; on coupe lorsque des groupes d'entités commencent à être fusionnées au lieu de fusionner des entités individuelles. Cela s'explique par notre objectif de réduire la redondance des attributs à un certain niveau, sans réduction excessive pouvant entraîner une perte d'information.

- Étape 3: Un critère d'évaluation approprié est appliqué pour choisir l'attribut le plus représentatif de chaque groupe/cluster. Dans le **FCRSC**, nous commençons par les deux ingrédients disponibles, c.-à-d., la solution de clustering obtenue par clustering hiérarchique et le vecteur de poids obtenu par **Relief-Sc**. Ensuite, pour chacun des clusters obtenus, le nombre d'attributs à l'intérieur est évalué. Lorsqu'un cluster est composé d'un seul attribut (considéré comme non redondant), il est directement ajouté au sous-ensemble d'attributs final choisi. Cependant, lorsque plusieurs attributs sont trouvés dans le même cluster, les attributs seront triés dans l'ordre décroissant de leur vecteur de pondération de marge correspondant. Ainsi, l'attribut avec le poids le plus élevé (le plus pertinent) est ajouté à l'ensemble d'attributs et les autres sont éliminés car ils sont jugés redondants. Après avoir obtenu les attributs représentatifs de chaque cluster, ceux-ci sont à nouveau triés dans l'ordre décroissant de leurs poids.

Afin de comparer les performances de nos méthodes suggérées avec d'autres méthodes de sélection d'attributs supervisés, non supervisés et contraintes basées sur les

scores, des expériences ont été réalisées sur des jeux de données de référence UCI et d'expression des gènes de grande dimension.

Contributions

Le cœur de cette thèse est la sélection active d'attributs semi-supervisée avec contraintes pour les données de grande dimension. Dans la mesure où la sélection d'attributs vise à trouver un petit sous-ensemble d'attributs pertinents et non redondants pouvant résumer avec succès l'espace d'attributs d'origine, notre travail devait se concentrer d'abord sur le problème de la sélection d'attributs pertinents avec contraintes, puis sur la gestion de la redondance. Tout au long de ce travail, nos contributions peuvent être résumées comme suit:

- Nous présentons une revue bibliographique et concise sur les algorithmes basés sur Relief à partir de leurs quatre interprétations possibles (probabiliste, compréhensible, mathématique et statistique), tout en soulignant leurs points forts, leurs limites, leurs variantes et leurs extensions, en plus de représenter l'algorithme Relief standard comme un algorithme basé sur la marge.

- Nous proposons des algorithmes de sélection d'attributs semi-supervisés basés sur les marges **Relief-Sc** (Relief avec contraintes de comparaisons) et sa version robuste nommée **ReliefF-Sc**. Ils intègrent la modification de l'hypothèse de marges quand elle est utilisée avec des contraintes cannot-link, avec la solution analytique de l'algorithme Relief supervisé du point de vue de son optimisation. Ils utilisent des contraintes cannot-link uniquement pour résoudre un problème

convexe simple dans une forme fermée offrant une solution unique.

- Nous suggérons une méthode active pour la sélection de contraintes par paires appelée Sélection de Contrainte Active (**ACS**). Le résultat de cette méthode est utilisé par **Relief-Sc** et **ReliefF-Sc**. **ACS** est basé sur la théorie de perturbations matricielles, et en particulier, sur le théorème de perturbation de vecteurs propres de premier ordre. L'algorithme choisit systématiquement les paires de données les plus efficaces pour réduire les incertitudes. En conséquence, seuls ces couples sont soumis à l'expertise humaine, ce qui permet de réduire le coût de la main-d'œuvre. En outre, une méthode de propagation de ces contraintes sélectionnées dans leur voisinage (**PACS**) a également été suggérée.

- Nous proposons d'étendre notre méthode de sélection d'attributs semi-supervisée à une nouvelle combinaison de regroupement d'attributs et de maximisation des marges d'hypothèses. Cette méthode appelée Feature Clustering ReliefF-Sc (**FCRSC**), permet d'éliminer la redondance dans le cadre de notre travail global suggéré.

Structure de la Thèse

Cette thèse est structurée comme suit:

Dans l'introduction générale, nous fournissons une brève introduction au problème de sélection d'attributs dans un contexte semi-supervisé. nous décrivons également les principales contributions de la thèse.

Dans le premier chapitre, nous présentons les définitions de réduction de dimensionnalité, d'extraction d'attributs, de sélection d'attributs, de pertinence et de redondance d'attributs. Nous présentons également les principales notations de données et la représentation des connaissances, ainsi que les méthodes de construction des données à base de graphes. En outre, nous classons le processus de sélection d'attributs en fonction de la disponibilité des informations de supervision (labellisation de classe et contraintes par paires) et en fonction du critère de performance de l'évaluation, tout en présentant l'ordonnancement des attributs selon leurs scores.

Dans le deuxième chapitre, nous présentons une bibliographie sur les algorithmes Relief de type filtre les plus populaires et nous mettons l'accent sur l'importance des marges dans ces algorithmes. L'algorithme Relief supervisé original est expliqué en détail en mettant l'accent sur ses points forts, ses points faibles et sur ses applications dans différents contextes en tant qu'algorithme sensible au contexte. Nous couvrons également toutes les variantes et extensions de Relief suggérées pour traiter les problèmes de données bruitées, incomplètes et à classes multiples. Le chapitre est divisé en quatre sections principales, chacune exprime une interprétation possible différente (probabiliste, compréhensible, mathématique et statistique).

Dans le troisième chapitre, nous proposons dans un premier temps l'algorithme **Relief-Sc** et sa version robuste **ReliefF-Sc**. Leur objectif est de réduire la haute dimensionnalité des données en trouvant un sous-ensemble d'attributs pertinents uniques,

dans un contexte semi-supervisé à l'aide de contraintes cannot-link. Par conséquent, nous expliquons d'abord le changement de principales notions de marge du contexte supervisé au contexte avec contraintes. Nous formulons également le **Relief-Sc** avec contraintes par l'interprétation mathématique des RBA. De plus, nous présentons l'algorithme basé sur les marges avec contraintes (Simba-Sc) principalement utilisé dans nos comparaisons des performances de **Relief-Sc**. D'autre part, nous présentons notre méthode **FCRSC** pour l'élimination ou la minimisation de redondance avec ses trois principaux blocs de construction: (1) la construction de graphes creux pour représenter les similarités d'attributs, (2) le regroupement hiérarchique sur ce dernier, (3) la combinaison de la maximisation des marges avec les résultats de la segmentation d'attributs, ce qui optimise la pertinence tout en minimisant la redondance. Enfin, nous validons expérimentalement l'efficacité de **Relief-Sc**, **ReliefF-Sc** et **FCRSC** sur plusieurs bases d'apprentissage automatique UCI et deux jeux de données d'expression de gènes de grandes dimensions par rapport aux méthodes de sélection supervisées, non supervisées et semi-supervisées.

Dans le quatrième chapitre, nous présentons nos méthodes de sélection et de propagation de contraintes actives en expliquant les travaux correspondants. En fait, nous avons divisé le chapitre en deux parties, la première explique notre contribution essentielle, c.-à-d., le processus de sélection des contraintes de comparaison à utiliser par l'algorithme de sélection d'attributs basé sur les marges avec contraintes, **Relief-Sc**. La seconde partie discute l'accroissement des informations de supervision en propageant ces contraintes appelées **PACS**. Enfin, des expériences approfondies ont été appliquées

sur des ensembles de données de référence UCI et sur deux systèmes d'expression de gènes de grandes dimensions pour valider les performances de nos méthodes, ainsi que pour montrer l'effet des contraintes générées aléatoirement (RCG) par rapport aux contraintes sélectionnées (**ACS**) dans le processus de sélection d'attributs avec contraintes.

Finalement, dans la conclusion générale, nous résumons nos constatations, contributions et limites et en proposant des perspectives pour la continuité de ce travail.