A Closest Resemblance Classifier with Feature Interval Learning and Outranking Measures for Improved Performance

Nabil Belacel *

Posted Date: 16 July 2024

doi: 10.20944/preprints202407.1257.v1

Keywords: Classification; Machine learning; Supervised learning; Feature Interval Learning; Outranking measures

 doi:10.20944/preprints202407.1257.v1

Article

A Closest Resemblance Classifier with Feature Interval Learning and Outranking Measures for Improved Performance

Nabil Belacel

National Research Council Canada; nabil.belacel@nrc-cnrc.gc.ca

Abstract: Classifiers face a myriad of challenges in today's data-driven world, ranging from overfitting and high computational costs to low accuracy, imbalanced training datasets, and the notorious black box effect. Furthermore, many traditional classifiers struggle with the robust handling of noisy and missing feature values. In response to these hurdles, we present classification methods that leverage the power of feature partitioning learning and outranking measures. Our classification algorithms offer an innovative approach, eliminating the need for prior domain knowledge by automatically discerning feature intervals directly from the data. These intervals capture essential patterns and characteristics within the dataset, empowering our classifiers with newfound adaptability and insight. In addition, we employ outranking measures to mitigate the influence of noise and uncertainty in the data. Through pairwise comparisons of alternatives on each feature, we enhance the robustness and reliability of our classification outcomes. The developed classifiers are empirically evaluated on several data sets from UCI repository and are compared with well-known classifiers including k Nearest Neighbors (K-NN), Support Vector Machine (SVM), Random Forest (RF), Neural Network (NN), Naive Bayesian (NB) and Nearest Centroid (NC). The experiments result demonstrate that the classifiers based on feature interval learning and outranking approaches are robust to imbalanced data and to irrelevant features and achieve comparably and even better performances than the well-known classifiers in some cases. Moreover, our proposed classifiers produce more explainable models whilst preserving high predictive performance levels.

Keywords: Machine learning; classification; supervised learning; feature interval learning; outranking measures

1. Introduction

Classification stands as a cornerstone in machine learning, tasked with assigning new data points to predefined classes. Commonly employed concept models such as decision trees (DTs), support vector machines (SVMs), and k-nearest neighbors (k-NNs) have long served this purpose [1]. Despite their prevalence, many existing classifiers encounter challenges, including overfitting, high computational costs, and a lack of interpretability—the infamous "black box" effect. Furthermore, these models often struggle to robustly handle noisy or missing feature values. Additionally, many existing classifiers rely on complex learning mechanisms by combining or transforming the data features, which can exacerbate issues such as overfitting, high computational costs, and a lack of interpretability—the so-called "black box" effect. Moreover, these models often grapple with effectively handling noisy or missing feature values. In response to these challenges, this paper devoted to classifiers based on feature interval learning (FIL), offering a distinct approach to traditional classification methods. FIL algorithms segment the feature space into distinct intervals for each feature, with each interval linked to a specific class. Feature intervals represent features as intervals rather than individual values, providing robustness to noise and variability in the input data, particularly for continuous data types [2]. A set of feature intervals embodies a concept on each feature dimension separately. Subsequently, each feature contributes to the classification process by distributing votes among classes, with the predicted class being the one receiving the highest vote [3]. This method's reliance on clear

 doi:10.20944/preprints202407.1257.v1

feature ranges offers interpretability, rendering decision rules easy to understand [2]. Furthermore, FIL can be effective with smaller datasets compared to data-hungry deep learning models [4]. In response to these challenges, we introduce an approach to supervised classification that harnesses the synergies of feature interval learning and outranking measures, called Closest Resemblance (*CR*). Outranking, entrenched in preference learning [5,6], transcends simple distance measures by autonomously learning feature intervals from data, thereby capturing nuanced patterns and enhancing adaptation to specific datasets. Moreover, outranking adeptly navigates noise and uncertainties in real-world data through pairwise comparisons on each feature. The proposed CR method amalgamates FIL's interpretability and efficiency with outranking's data-driven pattern recognition and robustness to noise. This integrated approach aims to surmount the limitations of existing methods, particularly in scenarios with imbalanced data and irrelevant features. Empirical evaluations conducted on diverse datasets from the UCI repository compare the proposed classifier CR against established classifiers such as k-Nearest Neighbors (k-NN), Support Vector Machines (SVM), Random Forests (RF), Multi-Layer Perceptron Neural Networks (MLP), Naive Bayes (NB), and Nearest Centroid (NC). The results underscore the efficacy of FIL with outranking, showcasing comparable or superior performance while delivering more interpretable models. This enhanced interpretability fosters trust and understanding in the decision-making process, bridging the gap between predictive power and internal workings.

2. Related Works

In the domain of supervised classification, various methodologies have been proposed to tackle the challenges posed by batch data and to enhance the interpretability and adaptability of classifiers. Among these, feature interval learning (FIL) techniques have emerged as prominent approaches. FIL approaches offer a distinct paradigm for classification, wherein the feature space is partitioned into intervals for each feature. Each interval is associated with a specific class, enabling a more nuanced representation of feature patterns and enhancing robustness to noise and variability in the data [2]. FIL methods encompass a range of techniques, including hypercuboid or hyperrectangle learning and prototype-based classification, which learn decision boundaries or representative prototypes to distinguish between classes. The concept of hypercuboid has been exploited for many years [7]

Hyper-cuboid approaches represent one category of FIL methods, where the feature space is partitioned into hyper-cuboid regions based on the distribution of training data [7]. These regions serve as decision boundaries, facilitating efficient classification of new instances. Hyper-cuboid approaches are particularly effective in handling high-dimensional data and are known for their computational efficiency and simplicity. Prototype-based FIL methods, on the other hand, generate prototypes for each class based on the distribution of feature intervals in the training data [3,8,9]. These prototypes encapsulate the characteristic features of each class and are used to classify new instances by measuring their resemblance to the prototypes. Prototype-based FIL methods offer interpretability and transparency in decision-making, as the classification process is based on clear feature ranges. In [2] describes a family of FIL algorithms. Some of FIL algorithms are voting feature intervals [3], k-nearest neighbor on feature projections [10] and Naive Bayes classifier (NBC) [1]. The common idea of the FIL algorithms, learn the concept description in the form of a set of disjoint intervals separately for each feature. At the testing phase the FIL algorithms predict the class of new data point as the label of with the highest total weighted votes of the individual features. For example in Figure 1, the horizontal axis is feature f_1 and the vertical axis is the feature f_2. The two rectangles represent the two prototypes of classes C^1 and C^2. The prototype of class C^1 is represented by the interval $I_{1,1}$ and $I_{2,1}$ (intervals determined using the training sets of class C^1 'the blue x sign'. And the prototype of class C^2 is represented by the intervals $I_{1,2}$ and $I_{2,2}$ (intervals calculated using the training sets of class C^2 'the green o sign'. The objective is to assign the unlabeled data (the red star $*$ in Figure 1) to the closest rectangular.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 July 2024 doi:10.20944/preprints202407.1257.v1

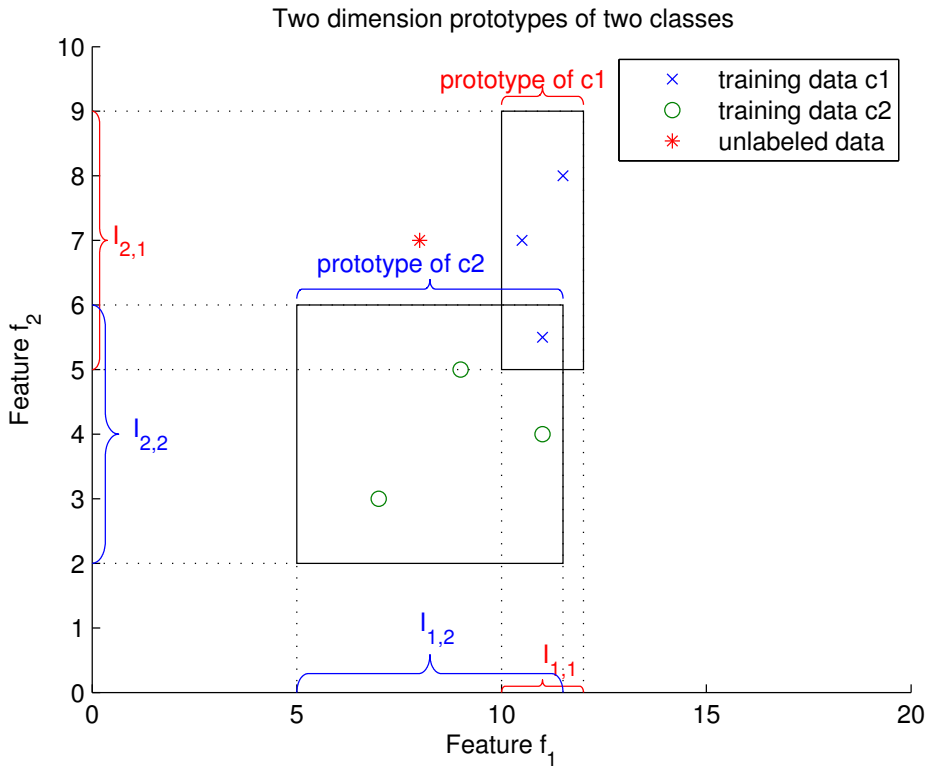

Figure 1. Two dimension prototypes of two classes in *CR* algorithm.

Overall, while hypercuboid or hyperrectangle learning algorithms offer computational efficiency and simplicity, FIL methods provide greater flexibility, robustness, and interpretability in classifying new data points, especially in scenarios with complex data distributions and high-dimensional feature spaces. However, predicting class labels based on the highest total weighted votes of individual features is a straightforward approach in FIL algorithms, it may have limitations in handling feature importance, noise, feature relationships, imbalanced datasets, and the trade-off between interpretability and accuracy. Addressing these inconveniences requires careful consideration of feature selection, noise reduction techniques, feature engineering, and robust mechanisms for handling imbalanced data. Beyond traditional majority voting approaches, another advancements in FIL is the integration of outranking measures into the classification process. Outranking measures, rooted in preference learning principles, extend the classification paradigm by capturing subtle preferences and trade-offs among alternative solutions [8,11]. By conducting pairwise comparisons on each feature, outranking measures facilitate more nuanced decision-making and enhance adaptability to specific datasets [5]. Several FIL classifiers based on outranking measures were developed since decades, among them we have PROAFTN [9], PROCFTN [12] and K Closest Resemblance (K-CR) or PROCTN [13]. These classifiers offer several advantages in the context of classification tasks. By evaluating the data points based on their pairwise performance on each feature, outranking techniques can capture subtle differences in feature importance and contribution to class membership. Instead of blindly aggregating votes from all features, outranking considers the relative performance of alternatives on each feature, effectively filtering out noise and focusing on discriminative features that contribute most to the classification decision. This robustness to noise enhances the reliability and accuracy of classification outcomes. Outranking techniques can capture complex relationships between features by conducting pairwise comparisons on each feature individually. Unlike majority voting, which treats features independently, outranking takes into account the interactions and dependencies between features, allowing for more accurate and nuanced classification decisions. Outranking techniques provide clear and interpretable decision rules by quantifying the resemblance relationship between data points.

 doi:10.20944/preprints202407.1257.v1

Instead of relying on a simple count of votes, outranking assigns a degree of resemblance between alternatives based on their pairwise comparisons, making the classification process more transparent and understandable. This enhanced interpretability fosters trust and understanding in the classification model's decision-making process. Outranking techniques can effectively handle imbalanced datasets by considering the relative performance of alternatives on each feature [14]. Instead of biasing the classification decision towards the majority class, outranking evaluates alternatives based on their overall resemblance relationship, ensuring that all classes are appropriately considered in the classification process. This helps mitigate the impact of class imbalance and ensures more balanced and accurate predictions. Outranking techniques offer flexibility and adaptability to different types of data and classification tasks. By capturing subtle differences in feature importance and performance, outranking can accommodate various data distributions and class structures, making it suitable for a wide range of classification scenarios. The FIL classifiers based on outranking measures have been applied to the resolution on many real world practical problems such as medical diagnosis [11,15–17], image processing and classification [14,18], engineering design [19], recommendation system [20] and cybersecurity [21]. Even though, these classifiers have several advantages, they are suffering from the computation complexity in their learning phase specifically when the number of features is very large. The PROAFTN and the K-CR build a set of prototypes in each class and then the inductive is based on the recursive approach, which is very complex. In this paper, we will introduce the FIL approach based on outranking measures for classification problems called Closest Resemblance *CR*. The *CR* algorithm is simple, effective, efficient and robust to irrelevant features. In the remaining paper, we will present the CR classifier, where each class is represented by one prototype and the classification rule is expressed as: "the data point is assigned to the class if the data point is resembled or roughly equivalent to the prototype of this class". Therefore, our proposed CR classifier is a prototype-based classification method that incorporates the FIL techniques to build the prototype of each class during the learning phase. The intervals in *CR* are inspired from the reference intervals used in the era of evidence-based medicine [22]. Unlike confidence intervals, which quantify uncertainty around a parameter estimate, prediction intervals capture the range within which a data point is likely to fall — they estimate the uncertainty associated with individual predictions.

3. The Closest Resemblance Algorithm

The *CR* is a supervised learning algorithm and it allows to calculate the resemblance measures by determining the preference relations. The rule of *CR* method in assigning a data points to a class is: "the data point s is assigned to a class if and only if s resemble or is (roughly) equivalent to the prototype of this class". Therefore, in order to assign point s to an appropriate class, *CR* will calculate the resemblance relation in order to measure the closeness of the sample s to the prototype of the class. To calculate the resemblance relation between the data point to assign and the prototype of the classes, *CR* is based on the preference relational system [5,6]. It employs a partial comparison between the sample and the prototypes of the classes, for each feature. Then, it applies a global aggregation using the concordance principle on the partial resemblance measure [5,8]. The *CR* classification method is considered as a weighted voting classifier in which each feature or attribute "votes" for the class membership of an unknown sample according to which class' prototype is the closest [8,9,13]. The partial comparison used by *CR* to calculate the resemblance between the sample and prototypes avoids resorting to conventional metric and non-metric distances that aggregate the score of all features in the same value unit. Hence, it helps to overcome some difficulties encountered when data is expressed in different units and to find the correct pre-processing and normalization data techniques. The algorithm *CR* proceeds into two phases:

3.1. Phase 1: Learning Phase

Suppose that a set of L labeled examples $X = \{x_1, x_2, ..., x_L\}$ with a set of n features $F = \{f_1, f_2..., f_n\}$ are given. This set of examples known as training set. Each example x_i is represented by

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 July 2024 doi:10.20944/preprints202407.1257.v1

a vector $f(x_i) =< f_1(x_i), f_2(x_i), ..., f_n(x_i) >, i = 1, ..., L$, where $f(x_i)$ is a n dimensional vector where each component $f_j(x_i)$ represents the value of the feature f_j for the training example x_i. The training examples have exactly one of K distinct class labels. In the learning phase, a prototype of each class is determined using the training sets X. For each class $C^h, h \in \{1, ..., K\}$, CR determines a prototype $P(C^h)$ from training set X. The prototypes are considered as good representative of their class and are described by the score of each of the n features. More precisely, for each feature f_j and for each class C^h, an interval I_j^h is determined. The interval I_j^h is the value range of feature f_j with respect to class C^h. A n-dimension prototype of each class $C^h, h = 1, ..., k$ is built as presented in Equation (1).

$$p(C^h) = \left\{ I(f_1, C^h), I(f_2, C^h), ...I(f_j, C^h), ..., I(f_n, C^h) \right\} \tag{1}$$

The value domain of each dimension f_j is the value range or interval $I_j^h = I(f_j, C^h) = \left[I^1(f_j, C^h), I^2(f_j, C^h) \right]$ where $I^1(f_j, C^h)$ is the lower bound and $I^2(f_j, C^h)$ is the upper bound interval of feature f_j for the class C^h.

In learning phase we have used two approaches to build the prototype of classes. The both approaches utilize prototypes defined by reference intervals for each feature in a class. In this work we have chosen the simple approaches to determine the interval of each feature for each class. The first one is based on the mean μ and the standard deviation σ and the second one is based on the percentiles. The reference intervals based on percentiles and the mean and standard deviation are non-parametric methods as they don't require any assumptions about the underlying distribution of the data.

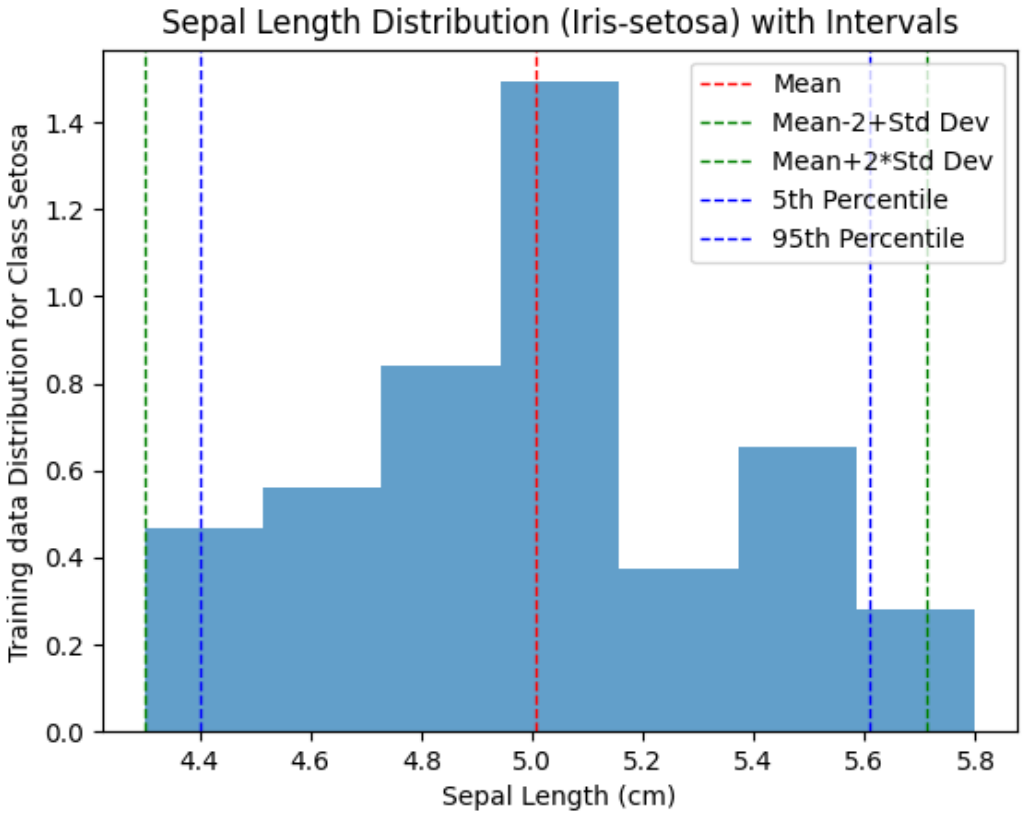

Figure 2. Reference intervals based on percentile and Mean and standard deviation.

3.1.1. Reference Intervals Based on Mean and Standard Deviation

To classify unlabeled data, the CR classifier determines the prototype of the classes by determining the reference interval of each feature based on the mean and standard deviation from the training set.

 doi:10.20944/preprints202407.1257.v1

Mean (μ) represents the "center" of the data distribution for a specific feature within a class. Standard deviation (σ) captures the spread of the data around the mean. They provide basic summaries of the data's central tendency (μ) and spread (standard deviation σ) without assuming a specific underlying distribution. This technique accounts for the spread of feature values within each class, creating intervals that extend $t > 0$ standard deviations from the mean. It adapts to the variability within each class.

For each class C^h and for each feature f_j, the mean μ and the standard deviation σ are calculated from the training data $x_i \in X and x_i \in C^h$ as presented in Equation (2).

$$I_j(f_j, C^h) = [\mu(f_j, C^h) - t * \sigma(f_j, C^h), \mu(f_j, C^h) + t * \sigma(f_j, C^h)] \tag{2}$$

.

By defining the interval as in Equation (2), we capture a range that encompasses a certain portion determined by t of the data points x_i in the class C^h for the feature f_j. Varying t allows control over the strictness of the interval. A higher t captures a wider range, a lower t captures a tighter range. This approach utilizes the mean and standard deviation to build the prototype of classes from the training set by leveraging Chebyshev's Theorem, we establish boundaries around the mean based on a chosen number t of standard deviations. Data points falling within these boundaries are considered typical, while those exceeding the boundaries are classified as outliers of the corresponding class. This method is particularly advantageous when the underlying data distribution is unknown, as Chebyshev's Theorem makes minimal assumptions about the data:

Chebyshev's Theorem: For any data distribution, at least $1 - (\frac{1}{t})^2$ proportion of the data points will fall within t standard deviations of the mean, where t is any positive constant.

Chebyshev's Theorem provides a lower bound to the proportion of measurements that are within a certain number of standard deviations from the mean.

 doi:10.20944/preprints202407.1257.v1

Algorithm 1 Prototype-Based Classification with reference Intervals: Training phase

Input: Training set; $t \geq 0$: Number of standard deviations to consider for interval width; L: Lower percentile and U: Upper percentile;
Output: a set of prototype $p(C^h), h = 1, ..., k$
Divide training set into classes and calculate intervals for each feature of each class:
for each class C^h **do**
  for each feature f_j **do**
    if Reference Interval==mean&Standard Deviation **then**
      Calculate the mean μ and standard deviation σ of f_j for training points in class C^h.
      Calculate the low boundary of the interval as $I_j^1(f_j, C^h) = \mu(f_j, C^h) - t * \sigma(f_j, C^h)$
      Calculate the high boundary of the interval as $I_j^2(f_j, C^h) = \mu(f_j, C^h) + t * \sigma(f_j, C^h)$
    else if Reference Interval==Percentile **then**
      Calculate the Lth and Uth percentiles of f_j for the training points in class C^h.
      Calculate the low boundary of the interval as $I_j^1(f_j, C^h) = percentileL(f_j, C^h)$
      Calculate the high boundary of the interval as $I_j^2(f_j, C^h) = percentileU(f_j, C^h)$
    end if
    Define the reference interval for feature f_j in class C^h as:
    $I_j(f_j, C^h) = [I_j^1(f_j, C^h), I_j^2(f_j, C^h)]$
    $P(C^h) = P(C^h) \cup I_j(f_j, C^h)$
  end for
  Create prototype profile $p(C_h) = [I(f_1, C_h), I(f_2, C_h), ..., I(f_n, C_h)]$
end for
return Set of prototypes $\left\{P(C^1), P(C^2), ..., P(C^h)\right\}$

3.1.2. Prototype-Based Classifiers with Percentile-Based Reference Intervals

This approach modifies the previous method by using percentiles to define reference intervals for each feature within a class.

For each class C^h and for each feature f_j, calculate the lower quartile L_j^h (ex. 25th percentile) and upper quartile U_j^h (ex. 75th percentile) using the training data (see Equation (3)).

$$I_j(f_j, C^h) = [L_j^h(f_j, C^h), U_j^h(f_j, C^h)] \tag{3}$$

Percentiles capture the distribution of data points within a class for each feature. The L_j^h and U_j^h percentiles define an interval that encompasses a specific proportion L% to U% of the data points in that class for that feature. This method is robust to outliers as extreme values have less influence on percentiles compared to mean and the standard deviation.

This technique captures the central tendency and spread of feature values within each class, offering a robust representation of the distribution by focusing on the lower and upper quartiles. Using percentile intervals for each feature of each class to represent a profile of the class in a classification problem can offer several benefits: By calculating percentile intervals for each feature within each class, create distinct profiles that characterize each class. This can help in understanding the unique characteristics and behaviors associated with each class in classification problem. Percentile intervals highlight the range of values that are most representative of each class. Features with wider intervals or larger differences between classes may indicate greater importance in distinguishing between classes. Class profiles based on percentile intervals are intuitive and easy to interpret. They provide clear boundaries that define the typical range of feature values associated with each class, making

 doi:10.20944/preprints202407.1257.v1

it easier to understand the characteristics of each class. Class profiles based on percentile intervals can aid in feature selection by identifying features that are most discriminative between classes. Features with significant differences in percentile intervals across classes may be more informative for classification tasks. When presenting classification results to stakeholders or end-users, class profiles based on percentile intervals provide a transparent explanation of how features contribute to class prediction. This enhances the interpretability and trustworthiness of the classification model. Similar to using percentile intervals for the entire training set, calculating class profiles based on percentile intervals can make the classification model more robust to outliers within each class. Outliers have less influence on percentile intervals compared to other summary statistics like mean and standard deviation. Class profiles based on percentile intervals can be visually represented using box plots or similar visualization techniques. These visual representations can offer a concise summary of the distribution of feature values within each class, facilitating comparison and interpretation. Overall, using percentile intervals to represent class profiles in a classification problem can enhance model interpretability, aid in feature selection, and provide valuable insights into the characteristics of each class, thereby improving the overall performance and understanding of the classification model. These techniques serve as crucial steps in learning the *CR* classification, offering different perspectives on how to define feature intervals. The Mean and Standard Deviation technique presented in Equation (2) adapts to class-specific variations, and the Percentile-based approach presented in Equation (3) provides a robust summary of the feature distributions within each class. Each technique contributes to the overall goal of capturing essential patterns and characteristics for the subsequent phases of the classification process. To learn the *CR* classifier we follow the different steps of the algorithm 1.

3.2. Phase 2: Classification Phase

The classification phase of *CR* classifier is outlined in Algorithm 2.

 doi:10.20944/preprints202407.1257.v1

Algorithm 2 Prototype-based classification with reference intervals: classification phase

Input: Test set $S = \{s_1, s_2, ..., s_m\}$: where each data point s_i is presented by the profile $f(s_i) = \{f_1(s_i), f_2(s_i), ...f_j(s_i), ..., f_n(s_i)\}$ where $f_j(s_i)$ is the value of the feature f_j for the sample s_i; $w_j : j = 1, ..., n$ a list of features' weights; A set of prototype:$P = \left\{P(C^1), P(C^2), ..., P(C^h), ..., P(C^K)\right\}$ obtained from algorithm 1
Output: Predicted class label for each test data point $s_i \in S$
for each sample s_i **do**
  for each class C^h **do**
    for each feature f_j **do**
      Calculate the partial distance: Equation (4)
    end for
  end for
  for each feature f_j **do**
    for each class C^h **do**
      for all Classes $C^t, t \neq h$ **do**
        Calculate partial resemblance: $R_j^{S_i}(p(C^h), p(C^t))$: Equation 5
      end for
    end for
  end for
  for Each class C^h **do**
    for all Classes $C^t, t \neq h$ **do**
      Calculate weighted resemblance matrix $R^{s_i}(p(C^h), p(C^t))$: Equation 6
    end for
  end for
  for Each class C^h **do**
    Calculate global score: Equation 9
  end for
  Predict class label for the sample s_i: Equation 10
end for
return Class labels for $x_i, i = \{1, 2, ...\}$

First for each testing data point s, CR determines a performance matrix of prototypes as presented in Table 1, where each component of the matrix corresponds to the absolute distance between the data point s to be classified and the prototype of the class according to the feature f_j. Each class $C^h, h = 1, .., k$ is represented by one prototype $p(C^h)$. The absolute distance between the prototype $p(C^h)$ of the class C^h and the data point s according to the feature f_j is calculated by the Equation (4). This function essentially measures the maximum deviation of the data point from the interval bounds. A value of 0 indicates perfect alignment, while higher values indicate greater deviation.

$$d_j^h(s, p(C^h)) = max\left\{0, I_{j,h}^1 - f_j(s)), f_j(s) - I_{j,h}^2\right\} \tag{4}$$

where $f_j(s)$ is the feature f_j value of the testing data point s. $p(C^h)$ is the prototype of the class C^h. The interval $\left[I^1(f_j, C^h), I^2 j(f_j, C^h)\right]$ presents the score of the prototype $p(C^h)$ according to feature f_j as presented in the learning phase in section 3.1

 doi:10.20944/preprints202407.1257.v1

Table 1. Performance matrix of prototypes for data point s

	f_1	.....	f_j	.....	f_n
$p(C^1)$	$d_1^s(p(C^1))$	.....	$d_j^s(p(C^1)$	.....	$d_n^s(p(C^1)$
$p(C^2)$	$d_1^s(p(C^2))$	.....	$d_j^s(p(C^2)$	.....	$d_n^s(p(C^2)$
.....	.....	.....	.....	.....	.....
$p(C^h)$	$d_1^s(p(C^h))$	.....	$d_j^s(p(C^h)$	.....	$d_n^s(p(C^h)$
.....	.....	.....	.....	.....	.....
$p(C^k)$	$d_1^s(p(C^k))$	.....	$d_j^s(p(C^k)$	.....	$d_n^s(p(C^k)$

Based on this performance matrix, the prototype nearest to the sample s is selected by calculating the weighted average mean of partial preference relationship between the sample s and the prototypes of different classes [8,13]. The preference relation between the different prototypes called also the outranking relation, can be defined as follows:

Definition. The prototype $p(C^h)$ outranks the prototype $p(c^l)$ "$p(C^h)R^s p(C^l)$ if and only if the resemblance between the sample s and the prototype $p(C^h)$ is stronger than the resemblance between s and the prototype $p(C^l)$ on the whole set of features. The outranking relation is based on the introduction of the partial outranking indices R_j^s [5,9]. Each index indicates weather the following statement is true or false: "The distance between the data point s and a given prototype is at least as good as the distance between data point s and the another prototype according to feature f_j". The R_j^s outranking index according to feature f_j is given by Equation (5).

$$R_j^s(p(C^h), p(C^l)) = \begin{cases} 1 & if \quad d_j^s(p(C^h)) \leqslant d_j^s(p(C^l)) \\ 0 & Otherwise \end{cases} \tag{5}$$

Let w_j denotes the relative importance coefficient of feature f_j, for all $f_j \in F$ (assume, without loss of generality $\sum_{j=1}^{n} w_j = 1$. This coefficient can be viewed as an intrinsic weight: it can be interpreted as the voting power of each feature. The higher the intrinsic weight, the more important the feature is. Note that the voting power in our classifier CR is independent on the value of the feature on the data points. From these partial feature-based outranking indices and by taking into account the relative importance of features w_j, we determine for each pair of prototypes $(p(C^h), p(C^l))$ the global outranking index $R^s(p(C^h), p(C^l))$, which is determined by the Equation (6).

$$R^s(p(C^h), p(C^l)) = \sum_{j=1}^{n}(w_j \times R_j^s(p(C^h), p(C^l))), h, l = 1, ..., K; j = 1, ..., n. \tag{6}$$

The $R^s(p(C^h), p(C^l))$ translates the degree with which the resemblance between the data point s and prototype of class C^h, $p(C^h)$ is stronger than the resemblance between the data point s and the prototype of the class C^l.

The CR classifier is based on a scoring function from an outranking relation R^s to choose the best prototype in terms of his resemblance with the data point s to be classified. Based on the outranking relations R^s between the prototypes of classes, CR selects the best prototype in terms of its resemblance with the data point to be classified. The scoring function is used to select a prototype, that is more closely to the data point s. The scoring function is inspired from the scoring function that used by the multicriteria decision analysis method PROMETHEE [23]. It allows to calculate the degree with which a prototype of class is preferred to all other prototypes and inversely how others are preferred to it.

 doi:10.20944/preprints202407.1257.v1

This is will be defined by calculating the globally positive outranking flow given by Equation (7) and negative outranking flow given by Equation (8).

$$\phi^{+}(p(C^{h})) = \sum_{l=1, l\neq h}^{K} R^{s}(p(C^{h}), p(C^{l})) \tag{7}$$

$$\phi^{-}(p(C^{h})) = \sum_{l=1, l\neq h}^{K} R^{s}(p(C^{l}), p(C^{h})) \tag{8}$$

The positive outranking flow ϕ^{+} defines the strength of the prototype $p(C^{h})$ regarding its resemblance with the data point s. The negative flow ϕ^{-} defines the weakness of the prototype $p(C^{h})$ regrading its resemblance with the data point s.From these two flows determined in Equation (7) and Equation (8), the net flow ϕ can be calculated by Equation (9).

$$\phi(p(C^{h})) = \phi^{+}(p(C^{h})) - \phi^{-}(p(C^{h})) \tag{9}$$

The net flow ϕ is the score of the prototype $p(C^{h})$ according to the outranking relation R^{s}. The first highest value of ϕ represents the best prototype regarding its resemblance with data point s, which means that the class C^{h} is the best candidate for the data point s. Hence, the CR classifies the unlabeled data point s by the following decision rule given by the Equation (10).

$$s \in C^{h} \Leftrightarrow \phi(p(C^{h})) = \max_{l\in\{1,\ldots,K\}} \left\{\phi(p(C^{l}))\right\} \tag{10}$$

The classification procedure used by CR to classify data to the preferred classes is summarized in the algorithm 2.

3.3. Complexity Analysis

This section is dedicated to the complexity analysis of our CR classifier in terms of space and time complexities. The complexity analysis is done on training process and classification of new single test point. we should denote by m, n and K the number of training set, number of features, and number of classes.

3.3.1. Space Complexity Analysis

Space Complexity in the training process: we need to store the intervals for each feature and each class. If we have n features and K classes, the space complexity is $O(n \times K)$ for storing these intervals.

Space complexity for the classification process for a single test example, we need to store the performance matrix, partial resemblance matrix, and weighted resemblance matrix. These matrices have dimensions $n \times K$, where n is the number of features and k is the number of classes. Therefore, the space complexity is $O(n \times K)$.

3.3.2. Time Complexity Analysis

Time complexity for the training process: Calculating the mean and standard deviation for each feature and each class requires iterating over all samples in the training set. Therefore, the time complexity is $O(m \times n \times K)$, where m is the number of samples. Creating the prototype profile for each class also requires iterating over all features for each class, resulting in a time complexity of $O(n \times K)$. Overall, the time complexity of the training process is dominated by the calculation of mean and standard deviation, resulting in $O(m \times n \times K)$.

Time Complexity for the classification process for a single test example, calculating the distance between the test example and each class prototype requires iterating over all features for each class, resulting in a time complexity of $O(n \times K)$. Calculating the partial resemblance matrix, weighted

 doi:10.20944/preprints202407.1257.v1

resemblance matrix, and general scores also involves operations with time complexity $O(n \times K)$. Overall, the time complexity of classifying a single test example is $O(n \times K)$. In summary, the training process has a space complexity of $O(n \times K)$ and a time complexity of $O(m \times n \times K)$, while the classification of a single test example has a space complexity of $O(n \times K)$ and a time complexity of $O(n \times K)$.

4. Experiments

The experiments were conducted on a PC with an Intel(R) processor running at 2.2 GHz and equipped with 2 processors and 32 GB of RAM. All classifiers and experiments in this research were implemented in Python, utilizing open-source libraries such as Scikit-learn [24] and Pandas [25]. The source code is available on GitHub at [GitHub link].

The real-world datasets used for benchmarking and performance evaluations are sourced from the UCI Machine Learning Repository, well-established in machine learning research [26]. Details regarding the datasets' descriptions and their dimensionalities are summarized in Table 2.

Table 2. Description of the data sets used in our experiments

Data Set	Attribute	Class	Instances
Dermatology	34	6	366
HeartDisease	13	5	303
Thyroid	20	2	720
Pageblocks	10	4	548
Breast Cancer	9	2	699
Dermatology	34	4	366
Newthyroid	5	3	215
Geobia	147	9	169
Glass	9	6	214
Leaf	14	30	340
Sonar	60	2	208
SPECTF	44	2	267

Our developed algorithms, termed *CR*, include *CR1*, which employs mean and standard deviation, and *CR2*, utilizing percentile-based approaches during the learning phase (Section 3.1). These classifiers are benchmarked against well-known algorithms: k Nearest Neighbors (k-NN) with k=3, Random Forest (RF) with 300 trees, Support Vector Machine (SVM), Multi-Layer Perceptron (MLP) neural network, Naive Bayesian (NB), and Nearest Centroid (NC).

To evaluate classification models, various metrics such as accuracy, precision, recall, and AUC (Area Under the Curve) are commonly used. Unlike accuracy, which solely measures correct predictions without considering the model's ability to rank positive instances higher than negatives, AUC provides a more comprehensive assessment. AUC evaluates the model's performance across different thresholds, making it particularly effective for imbalanced datasets or scenarios where prioritizing true positives is crucial [27].

To ensure robust evaluation, we employed 5-fold cross-validation. This technique divides the dataset into five folds, using four for training and one for validation in each iteration. By averaging performance metrics such as AUC across these folds, we obtain a more reliable estimate of each classifier's generalization ability and mitigate the risk of overfitting to specific training data.

 doi:10.20944/preprints202407.1257.v1

4.1. Results and Discussion

Table 3 presents the classification performance of the *CR1*, *CR2*, K-NN, RF, SVM, MLP, NB, and NC classifiers. The results are based on the average AUC obtained from five repetitions of 5-fold cross-validation. For the *CR1* classifier, which utilizes mean and standard deviation-based reference intervals, we applied $t = 1.1$ standard deviations. For the *CR2* classifier, which employs percentile-based reference intervals, we used the 90th percentile ($U = 0.9$) as the upper bound and the 10th percentile ($L = 0.1$) as the lower bound. The best classification performance for each dataset is highlighted in boldface.

Based on the experimental study presented in Table 3, *CR1* (percentile-based) achieved the highest average AUC, closely followed by RF and *CR2* (mean and standard deviation-based). Tables 4 and 5 provide a robust analysis of the comparative performance of the developed classifiers *CR1* and *CR2* against other classifiers, based on mean AUC and average normalized AUC scores. As observed, both *CR1* and *CR2* consistently outperform other classifiers.

The classifiers can be categorized into three groups based on their average normalized AUC scores, as presented in Table 5:

- **Best approaches**: CR1, CR2, RF, and NB.
- **Middle approaches**: SVM, K-NN, and MLP.
- **Weakest approach**: NC.

Table 3. The performance of all classifiers based on the AUC average using 5-cross validation

Dataset	CR1(0.9-0.1)	CR2(t=1.1)	K-NN	RF	SVM	MLP	NB	NC
Dermatology	0.955	**0.983**	0.934	0.979	0.977	0.981	0.923	0.731
HeartDisease	0.605	**0.670**	0.505	0.580	0.605	0.600	0.618	0.562
Thyroid	0.909	0.763	0.551	**0.919**	0.501	0.505	0.475	0.598
Pageblocks	**0.883**	0.857	0.750	0.779	0.802	0.849	0.768	0.593
BreastCancer	0.969	**0.973**	0.971	0.967	0.960	0.960	0.963	0.956
Shuttle	0.917	0.929	0.885	0.926	0.917	**0.967**	0.872	0.720
Newthyroid	0.946	0.942	0.899	0.943	**0.987**	0.737	0.951	0.865
Geobia	**0.916**	0.895	0.692	0.910	0.752	0.708	0.866	0.658
Glass	0.748	0.719	0.789	**0.855**	0.743	0.604	0.686	0.718
Leaf	0.868	0.862	0.777	**0.876**	0.745	0.753	0.843	0.765
Sonar	**0.840**	0.751	0.815	0.825	0.759	0.806	0.698	0.667
SPECTF	0.696	0.525	0.616	0.627	0.654	0.619	0.761	**0.772**
Average AUC	**0.854**	0.823	0.766	0.849	0.784	0.7489	0.786	0.717
Average rank	**0.631**	0.624	0.515	0.622	0.553	0.473	0.608	0.363

 doi:10.20944/preprints202407.1257.v1

Table 4. Algorithm Rankings based on Average AUC Normalized Scores

Algorithms Ranking	Average Normalized Scores
CR1 (Percentile:0.9-0.1)	**0.631729**
CR2 (t=1.1)	**0.62406**
RF	**0.622963**
NB	0.608517
SVM	0.553853
k-NN	0.515612
MLP	0.473611
NC	0.363622

Table 5. Mean AUC rankings

Classifier	Mean Rank
CR1(Percentile:0.9-0.1)	2.692308
CR2(t=1.1)	3.153846
RF	3.230769
SVM	4.615385
NB	4.846154
MLP	5.307692
K-NN	5.538462
NC	6.615385

Based on the average rank presented in Table 3, our prototype-based approaches outperform other prototype-based classifiers, including K-NN, NB, and NC. Notably, *CR1* (percentile-based) achieved the highest average AUC among all classifiers.

Overall, using outranking techniques instead of majority voting provides several advantages, including robustness to noise, consideration of feature relationships, enhanced interpretability, handling of imbalanced datasets, and flexibility in adapting to different types of data. These advantages make outranking techniques a powerful and versatile tool for classification tasks, particularly in complex and challenging real-world scenarios.

Comparison between CR-Based Mean-Std and CR-Based Percentile Intervals

Mean and standard deviation are more sensitive to outliers, while percentiles are more robust. Mean-based intervals might not accurately capture skewed distributions. Percentile intervals are easier to interpret as they directly represent the proportion of data points within the interval. Choosing the appropriate method depends on the data characteristics and the importance of outlier handling. For datasets with outliers or skewed distributions, percentile-based reference intervals offer a more robust alternative to mean and standard deviation in this prototype-based classification approach.

In this study, we assumed equal weights for all features, which may not fully capture the varying degrees of importance each feature may have in classification tasks. In future research, we intend to explore optimization methods to assign appropriate weights to features dynamically. This approach aims to enhance classifier performance by giving more prominence to informative features while reducing the impact of less relevant ones, thereby potentially improving overall accuracy and robustness. Additionally, leveraging feature weight optimization for feature selection purposes could further refine our classifiers, focusing on the most discriminative features for each dataset. This paper utilizes inductive learning to establish the classification method *CR* and its parameters. The feature

 doi:10.20944/preprints202407.1257.v1

intervals are learned directly from the training data using inductive techniques. To mitigate the risk of overfitting, a deductive learning approach could be utilized, leveraging expert knowledge to fine-tune the intervals and set the feature weights appropriately [16]. To accelerate the *CR* learning process, parallel computation techniques can be employed. Utilizing different GPUs to handle each class independently during the prototype-building phase can significantly enhance processing speed and efficiency.

5. Conclusions

This work represents a significant advancement in classification methodologies, aiming to enhance both the practicality and interpretability of machine learning models in complex real-world scenarios. By integrating Feature Interval Learning (FIL) and outranking measures, we propose a novel approach in the form of Nested Generalized Exemplars classifiers. These classifiers leverage outranking techniques and utilize feature projections of training samples to encapsulate induced classification knowledge. New sample classification is achieved through weighted majority voting of attribute predictions.

Our empirical evaluation on various datasets from the UCI repository compares our approach with established classifiers such as k Nearest Neighbors, Support Vector Machine, Random Forest, Neural Network, Naive Bayesian, and Nearest Centroid classifiers. The results demonstrate that our FIL-based classifiers with outranking approaches exhibit robustness to imbalanced data and irrelevant features, achieving comparable or superior performance across most experiments. Furthermore, these classifiers yield more interpretable models while maintaining high predictive accuracy levels.

Funding: This research was funded by Digital Research Center of National Research Council Canada.

Data Availability Statement: The datasets and Python codes of *CR* classifier with the algorithms used in our comparative study are available on GitHub at: https://github.com/nbelacel/Closest-Resemblance

.

Acknowledgments: I would like to express my gratitude to our co-op students Rani Adhaduk and Durga Prasad Rangavajjala for their invaluable assistance in python implementation

Conflicts of Interest: The author declares no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

FIL	Feature Interval Learning
CR	Closest Resemblance classifier
CR1	CR uses reference interval based on mean and standard deviation
CR2	CR uses reference intervals based on percentile
DT	Decision Trees
SVM	Support Vector Machine
RF	Random Forest
NB	Naive Bayesian
NC	Nearest Centroid

References

1. Mitchell, T.M. Machine learning and data mining. *Communications of the ACM* **1999**, *42*, 30–36.
2. Dayanik, A. Learning feature-projection based classifiers. *Expert Systems with Applications* **2012**, *39*, 4532–4544. doi:https://doi.org/10.1016/j.eswa.2011.09.133.
3. Demiröz, G.; Güvenir, H.A. Classification by voting feature intervals. European Conference on Machine Learning. Springer, 1997, pp. 85–92.
4. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics; Teh, Y.W.;

Titterington, M., Eds.; PMLR: Chia Laguna Resort, Sardinia, Italy, 2010; Vol. 9, *Proceedings of Machine Learning Research*, pp. 249–256.

5. Roy, B. *Multicriteria Methodology for Decision Aiding*; Nonconvex Optimization and Its Applications, Springer US, 2013.
6. Vincke, P. *Multicriteria decision-aid*; John Wiley & Sons, 1992.
7. Salzberg, S. A nearest hyperrectangle learning method. *Machine learning* **1991**, *6*, 251–276.
8. Belacel, N. Méthodes de classification multicritère: Méthodologie et application à l'aide au diagnostic médical. PhD thesis, Univ. Libre de Bruxelles, Belgium, Brussels, 1999.
9. Belacel, N. Multicriteria assignment method PROAFTN: methodology and medical application. *European Journal of Operational Research* **2000**, *125*, 175–183.
10. Akkus, A.; Güvenir, H.A. K nearest neighbor classification on feature projections. Proceedings of the Thirteenth International Conference on International Conference on Machine Learning; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1996; ICML'96, p. 12–19.
11. Belacel, N.; Cuperlovic-Culf, M. PROAFTN Classifier for Feature Selection with Application to Alzheimer Metabolomics Data Analysis. *International Journal of Pattern Recognition and Artificial Intelligence* **2019**, *33*, 1940013. doi:10.1142/S0218001419400135.
12. Belacel, N.; Boulassel, M.R. Multicriteria fuzzy classification procedure PROCFTN: methodology and medical application. *Fuzzy Sets and Systems* **2004**, *141*, 203 – 217.
13. Belacel, N. The k-closest resemblance approach for multiple criteria classification problems. In *Modelling, Computation and Optimization Information and Management Sciences*; Hoai, L.; Tao, P., Eds.; Hermes Sciences Publishing, 2004; pp. 525–534.
14. Belacel, N.; Duan, C.; Inkpen, D. The K-Closest Resemblance Classifier for Remote Sensing Data. Canadian Conference on Artificial Intelligence, 2020, pp. 49–54.
15. Belacel, N.; Boulassel, M.R. Multicriteria fuzzy assignment method: a useful tool to assist medical diagnosis. *Artificial intelligence in medicine* **2001**, *21*, 201–207.
16. Belacel, N.; Vincke, P.; Scheiff, J.M.; Boulassel, M.R. Acute leukemia diagnosis aid using multicriteria fuzzy assignment methodology. *Computer Methods and Programs in Biomedicine* **2001**, *64*, 145–151.
17. Belacel, N.; Wang, Q.; Richard, R. Web-integration PROAFTN methodology for acute leukemia diagnosis. *Telemedicine Journal & e-Health* **2005**, *11*, 652–659.
18. Al-Obeidat, F.; Al-Taani, A.T.; Belacel, N.; Feltrin, L.; Banerjee, N. A Fuzzy Decision Tree for Processing Satellite Images and Landsat Data. *Procedia Computer Science* **2015**, *52*, 1192–1197. doi:https://doi.org/10.1016/j.procs.2015.05.157.
19. Sassi, I.; Belacel, N.; Bouslimani, Y. Photonic-crystal fibre modeling using fuzzy classification approach. *Int. J. Recent Trends Eng. Technol* **2011**, *6*, 100–104.
20. Belacel, N.; Wei, G.; Bouslimani, Y. he k Closest Resemblance Classifier for Amazon Products Recommender System. ICAART, 2020, pp. 873–880.
21. Al-Obeidat, F.; El-Alfy, E.S. Hybrid multicriteria fuzzy classification of network traffic patterns, anomalies, and protocols. *Personal and Ubiquitous Computing* **2019**, *23*, 777–791.
22. Ceriotti, F.; Hinzmann, R.; Panteghini, M. Reference intervals: the way forward. *Annals of Clinical Biochemistry* **2009**, *46*, 8–17, [https://doi.org/10.1258/acb.2008.008170]. PMID: 19103955, doi:10.1258/acb.2008.008170.
23. Brans, J.P.; Mareschal, B. PROMETHEE methods. In *Multiple criteria decision analysis: state of the art surveys*; Springer, 2005; pp. 163–186.
24. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; others. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* **2011**, *12*, 2825–2830.
25. McKinney, W.; others. Data structures for statistical computing in python. Proceedings of the 9th Python in Science Conference. Austin, TX, 2010, Vol. 445, pp. 51–56.
26. Dua, D.; Graff, C. UCI Machine Learning Repository, 2017.
27. Ling, C.X.; Huang, J.; Zhang, H. AUC: A Better Measure than Accuracy in Comparing Learning Algorithms. Advances in Artificial Intelligence; Xiang, Y.; Chaib-draa, B., Eds.; Springer Berlin Heidelberg: Berlin, Heidelberg, 2003; pp. 329–341.

 doi:10.20944/preprints202407.1257.v1