UNIVERSIDADE FEDERAL FLUMINENSE

ÉRICK OLIVEIRA RODRIGUES

Introducing Mathematical Morphology in Machine Learning

NITERÓI

2017

UNIVERSIDADE FEDERAL FLUMINENSE

ÉRICK OLIVEIRA RODRIGUES

Introducing Mathematical Morphology in Machine Learning

Thesis presented to the Graduate School of Computer Science of Universidade Federal Fluminense as a partial requirement for obtaining the Doctor of Science degree in Computer Science. Field: Visual Computing

Advisor:
AURA CONCI

NITERÓI

2017

Érick Oliveira Rodrigues

Introducing Mathematical Morphology in Machine Learning

Thesis presented to the Graduate School of Computer Science of Universidade Federal Fluminense as a partial requirement for obtaining the Doctor of Science degree in Computer Science. Field: Visual Computing.

Approved in December, 2017.

EXAMINERS

Prof. Dr. Aura Conci - Orientadora/Advisor, UFF

Prof. Dr. Luiz Satoru Ochi, UFF

Prof. Dr. José Viterbo Filho, UFF

Prof. Dr. Panos Liatsis, Khalifa University - UAE

Prof. Dr. Raul Queiroz Feitosa, PUC-RJ

Prof. Dr. Alexandre Gonçalves Evsukoff, UFRJ

Niterói

2017

"It is through science that we prove, but through intuition that we discover."

Henri Poincaré

Resumo

Esta tese introduz conceitos da morfologia matemática, um campo já estabelecido dentro da computação visual, ao aprendizado de máquina. Técnicas da morfologia matemática exploram características de forma e densidade, aspectos que não são devidamente explorados por algoritmos de machine learning tradicionais. Duas implementações de classificadores são apresentadas conjuntamente ao framework teórico proposto. Na classificação, a ideia principal é rotular o espaço ao redor das instâncias de treino, crescendo esse rotulamento a cada iteração. Nas implementações propostas, este crescimento respeita uma nova métrica/distância que também é proposta nesta tese. Esta métrica é mais rápida que as distâncias de Manhattan e Euclidiana nas iterações de vizinhanças discretas. Uma vez que o espaço de classificação encontra-se razoavelmente rotulado, este espaço é utilizado como modelo para classificar instâncias não rotuladas. Os classificadores propostos obtiveram melhores resultados em relação a 14 outros classificadores em 5 de 8 bases de dados, o que é uma forte evidência inicial do poder de predição da proposta, que é inovadora e ainda está em seus passos iniciais de desenvolvimento. Além disso, um algoritmo de clusterização que é baseado na reconstrução morfológica também é proposto. Neste caso, o algoritmo é mais rápido que o estado-da-arte ao passo em que provê resultados onde formas e densidade dos clusters também são preservadas. O algoritmo de clusterização possui características únicas como um senso intrínseco de clusters máximos que podem ser criados, uma forma de remover ruído das bases sem custos extra de processamento, permite alterações no padrão de crescimento dos clusters ao modificar os elementos estruturantes, entre outras. Um experimento considerando mais de 70 voluntários também demonstrou que clusterizações realizadas com algoritmos sensíveis às formas e densidade são mais próximos de clusterizações humanas.

Palavras-chave: classificadores morfológicos; aprendizado de máquina; aprendizado supervisionado; aprendizado não supervisionado; clusterização; morfologia matemática; dilatação de imagens; k-NN; teoria dos conjuntos; densidade de clusters; forma de clusters.

Abstract

This thesis introduces concepts from mathematical morphology, an established field in visual computing, to the field of machine learning. Mathematical morphology operations are sensitive to shapes and density, aspects that are not sufficiently exploited in usual machine learning techniques. Two implementations of classifiers that use these techniques are proposed along with a theoretical framework. In classification, the main concept is to label the space around the training instances, growing this labelling at each iteration. The growth conforms to a new distance metric proposed herein, which is substantially faster than Manhattan and Euclidean distances when it comes to iterations of discrete neighbourhoods. At last, these labelled spaces are used to classify unlabelled instances. The proposed classifiers outperformed 14 classifiers in 5 out of 8 datasets, which is a strong initial evidence of the predictive power of the approach, which is novel and is in its early stages of development. Furthermore, a clusterization algorithm that is based on morphological reconstruction is proposed. In this case, the algorithm is faster than state-of-the-art techniques while providing clusterization results that preserve shapes and cluster density. The proposed clusterization scheme possesses a number of unique features such as an intrinsic sense of maximal clusters that can be created, provides a means of removing noise from datasets with no extra processing costs, enables a vast amount of growth patterns, which are controlled by structuring elements, etc. An experiment with more than 70 volunteers also indicated that clustering algorithms that are sensitive to shapes and density produce more human-like clusterizations.

Keywords: morphological classifier; machine learning; data mining; supervised learning; unsupervised learning; clusterization; mathematical morphology; image dilation; k-nn; set theory; k-means; cluster density; cluster shape.

List of Figures

List of Tables

List of Acronyms

MM	:	Mathematical Morphology;
k-NN	:	k-Nearest Neighbours;
BMCA	:	Binary Morphological Clustering Algorithm;
MkNN	:	Morphological k-Nearest Neighbours;
MDC	:	Morphological Dilator Classifier;
k-MS	:	k-Morphological Sets;
GRASP	:	Greedy Randomized Adaptive Search Proble;
2D	:	2-Dimensional;
UCI	:	UC Irvine;

List of Symbols

k	:	Represents the amount of clusters or the amount of nearest neighbours (input parameter of the k-NN algorithm).
A	:	Represents an input image or a partition of an image.
a	:	In the Section 2, it represents elements of the input image A. In Section 2.2, it represents the attributes of a dataset.
B	:	Represents structuring elements.
b	:	Represents elements of structuring element B.
dil^B	:	Notation for the dilation operation using structuring element B.
er^B	:	Notation for the erosion operation using structuring element B.
$A(a)$	:	Returns the grey level value of image A at position or coordinate a.
op^B	:	Notation for the opening operation using structuring element B.
cl^B	:	Notation for the closing operation using structuring element B.
wth^B	:	Notation for the white top-hat operation using structuring element B.
bth^B	:	Notation for the black top-hat operation using structuring element B.
M	:	Mask that is used in geodesy to limit the range of morphological operators.
$R_M^B(A)$	:	Notation for the morphological reconstruction using structuring element B and mask M.
x	:	Represents instances of a problem (usually labelled instances).
p	:	Represents the number of dimensions or the p parameter of the Minkowski distance (in Section 2.4).
n	:	Usually represents the number of elements. In Section 2.4 it represents the number of dimensions.
l	:	Represents labels or classes.
L	:	Represents the total amount of labels in a problem.
c(x)	:	Returns the known label of instance x.
y	:	Represents instances of a problem (usually unlabelled instances).
P	:	Prototype sets.

arg	:	Argument of a function.
C	:	Represents clusters.
u	:	Cluster centroid.
d	:	Represents distances.
d_1	:	Represents the Manhattan distance.
d_2	:	Represents the Euclidean distance.
d_p	:	Represents the Minkowski distance.
d_∞	:	Represents the Chebyshev distance.
d_{SD}	:	Represents the Squared Euclidean distance.
d_{CAD}	:	Represents the Canberra distance.
P_{err}	:	Represents the classification error.
t_{err}	:	Represents a predefined misclassification threshold.
$\hat{c}$(y)	:	Function used to classify the unlabelled instance y.
X	:	Represents the training dataset.
Y	:	Represents a dataset of unlabelled instances (the test dataset, or the dataset that contains instances whose classification is pending).
i	:	Represents iterations.
q	:	Used to indicate the complement set partition.
$\mathbb{S}$	:	Classification space.
ϕ	:	Convergence function.
γ	:	Weights applied to the original instances of training datasets.
σ	:	Represents the maximal number of iterations.
T	:	Represents counters (voting counters) for each class or label of a given problem.
β	:	Similar to a structuring element, it represents a set of rules that indicates the orientation in which instances are dilated.
τ	:	Scaling or weighting factor.
W	:	Width of the input image.
H	:	Height of the input image.
O	:	Big-O notation.
s	:	Scaling factor.
h	:	Represents the coordinates of p-dimensional elements.
r	:	Represents the number of rectangles.
Θ	:	Represents the population of the evolutinary algorithm.

$\Psi()$	:	Generates a random number.
ε	:	Represents individuals in the evolutionary algorithm.
Π	:	Represents a maximal value/ceil for parameters (a predefined threshold).
G	:	Represents the grid or the discretized/quantized classification/clusterization space.
δ	:	Variable used to increase the size of the structuring element.

Contents

Chapter 1

Introduction

Mathematical Morphology (MM) uses set theory to analyse and process graphical objects. Some of its mostly famous operations alter their size, shape and convexity. MM operations are commonly defined as set operations. A dilation, for instance, uses a structuring element (also a set) to displace another set by summing up each pair of elements of both sets and unites the outcome to the current configuration.

Mathematical morphology is commonplace in the field of visual computing. Unfortunately, this is not the case for machine learning. Some works use a fairly limited amount of MM operations but are not entirely based on mathematical morphology, they use heuristics when it comes to connecting instances to their associated cluster. To the best of our knowledge, no work has ever proposed any classification technique that relies nor uses mathematical morphology techniques in classification. Thus, classification and clusterization methods that adhere to mathematical morphology are proposed herein.

Machine learning is well established and is vastly used in the fields of computer science and engineering, and to a great extent of subareas, ranging from robotics [23], computer vision [79], data mining, analysis and knowledge discovery [27], health care [83] and many others [66, 104]. Supervised learning (which includes classification) requires data of actual past situations for training algorithms, which are used to predict the outcome, or the label, of new situations. In this type of learning, the collected data must contain a label or a class. In other words, supervised learning consists of finding patterns in order to generate a predictive model that guesses the label of instances whose label is unknown.

Unsupervised learning (which includes clustering), on the other hand, focuses on grouping data respecting a similarity measure. In this case, no label is known during the training phase. This implies that the only data available for training are the instance at-

tributes. The algorithm must be intelligent enough to group and separate these instances based on heuristics and some sort of similarity analysis. The input data usually consists of several unlabelled instances, and the algorithm outputs a cluster label for each input instance.

1.1 Publications and Motivation

Through the course of this PhD research, we have been publishing studies using mathematical morphology and related techniques in image processing (in order of importance regarding the relationship with the proposal and development of this thesis):

1. Rodrigues, E.; Morais, F.; Morais, N.; Conci, L.; Neto, L.; Conci, A. A novel approach for the automated segmentation and volume quantification of cardiac fats on computed tomography. *Computer Methods and Programs in Biomedicine* (2015)

2. Rodrigues, E. O.; Pinheiro, V. H. A.; Liatsis, P.; Conci, A. Machine learning in the prediction of cardiac epicardial and mediastinal fat volumes. *Computers in Biology and Medicine* (2017)

3. Rodrigues, E. O.; Rodrigues, L. O.; Oliveira, L. S. N.; Conci, A.; Liatsis, P. Automated recognition of the pericardium contour on processed ct images using genetic algorithms. *Computers in Biology and Medicine 87* (2017), 38–45

4. Rodrigues, E. O.; Conci, A.; Morais, F. F. C.; Perez, M. G. Towards the automated segmentation of epicardial and mediastinal fats: A multi-manufacturer approach using intersubject registration and random forest. *IEEE International Conference on Industrial Technology (ICIT)* (2015), 1779–1785

5. Rodrigues, E. O.; Morais, F. F. C.; Conci, A. On the automated segmentation of epicardial and mediastinal cardiac adipose tissues using classification algorithms. *MEDINFO 2015: EHealth-enabled Health: Proceedings of the 15th World Congress on Health and Biomedical Informatics 216* (2015)

6. Rodrigues, E. O.; Viterbo, J.; Conci, A.; McHenry, T. A context-aware middleware for medical image based reports an approach based on image feature extraction and association rules. *IEEE International Conference on Computer Systems and Applications* (2015)

7. Rodrigues, E. O.; Porcino, T. M.; Conci, A.; Silva, A. C. A simple approach for biometrics: Finger-knuckle prints recognition based on a sobel filter and similarity measures. *International Conference on Systems, Signals and Image Processing (IWSSIP)* (2016)

8. Rodrigues, E. O.; Clua, E. A real time lighting technique for procedurally generated 2d isometric game terrains. *Entertainment Computing - ICEC 2015 9353* (2015), 32–44

Most of these works use machine learning techniques coupled with image processing techniques to achieve a satisfiable graphical solution. In this process, we noticed that visual computing vastly uses techniques and algorithms from machine learning, while the opposite is not quite true.

By investing on the idea of bringing concepts from visual computing to machine learning, we developed a theoretical framework and implementations for classification and clustering that are based on mathematical morphology. The initial approach was a paper on clusterization, published in a reputable journal:

9. Rodrigues, E. O.; Torok, L.; Liatsis, P.; Viterbo, J.; Conci, A. k-ms: A novel clustering algorithm based on morphological reconstruction. *Pattern Recognition 66* (2016), 392–403

In this pattern recognition work, the objective was to explore the capabilities inherit from MM of recognizing shapes and cluster density. Traditional clustering methods rely on the spatial dispositions, disregarding density.

Let us suppose that we want to separate the input data shown in Figure 1.1-(a) in two clusters. Most clusterization algorithms would either produce the results shown in (b) or (c). Although this reasoning makes sense in terms of spatial disposition, it disregards density. The two points in the middle are closely related, i.e., they are dense. In density-based clusterizations these aspects are positively exploited.

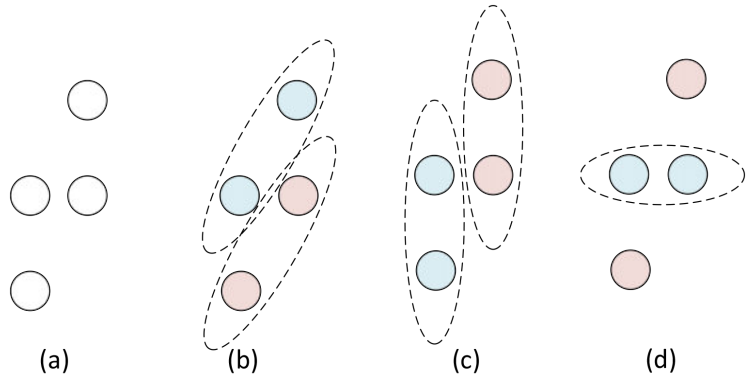


Figure 1.1: Clusterization results for an arbitrary configuration.

A possible density-based solution is the one shown in Figure 1.1-(d), where, in this case, the two data points at the center belong to the same cluster and the remaining is placed in a complement cluster or are treated as noise. The clusterization method proposed in this work uses morphological reconstruction to produce an actual density-based clusterization similar to this.

The proposed algorithm ended up being more efficient than state-of-the-art methods while also being very unique. The following list of features inherent to the proposed clustering technique is as follows:

1. Sense of maximal clusters that can be created (regardless of the k parameter), where the k parameter fulfils the same role as k-Means, i.e., limiting the amount of clusters.
2. Computationally efficient algorithm as it is amenable to parallelization.
3. Uses structuring elements that alter the way clusters are formed and grouped.
4. Provides a means for removing noise without introducing extra processing costs.

Later, we focused on bringing mathematical morphology to the classification paradigm. Once again, the premiss was to exploit shapes and density information, which are not sufficiently exploited by traditional classifiers (more details in Section 2.2.1). Furthermore, we were inspired by the fact that datasets reflect nature, and therefore reproduce fractal behaviour. Let us suppose that the points shown in Figure 1.2 (circles at the leaves of the tree) illustrate points of a given class. In this case, it is possible to model this information using a fractal function, represented by the tree.

Figure 1.2: Fractal data disposition.

Mathematical morphology is capable of modelling this information. A set can be grown in a fractal tree-like fashion (using dynamic structuring elements that adapt at each iteration) until it reaches a model that accurately encapsulates the data points shown in Figure 1.2. Some classifiers use fractal features, fractal analyses, and other related techniques, but these are not present during or as part of their classification model construction.

In summary, we provide a theoretical framework that is capable of dealing with fractal information, shapes and density. Section 2.2.1 describes other features of the proposed framework, such as the modelling of translations. Furthermore, produced classification models also stem from the features of mathematical morphology such as the generation of different results depending on the structuring element/fashion of growth of the model.

A possible approach in constructing classifiers that use dilations is starting with a set of training instances whose labels are known and growing them, filling the space around these instances in order to cover a wider range of unlabelled instances. As these instances grow, intersections will occur and either the growth stops or a voting scheme should be responsible for choosing the correct label when intersections occur.

Implementing these classifiers brought us to another contribution. We started coding our classification algorithms in Graphics Processing Units (GPU) to exploit the parallel potential of the methodology. That is, assuming the instance space is a p-dimensional grid, positions of the grid near positions that contain instances are iteratively marked as the model grows. Each grid cell can be grown in parallel, and therefore, as the GPU paradigm relies on processing the same instruction in parallel with different data, it turns

out to be suited for growing several grid positions at once, reducing processing times.

We head towards another issue. The neighbourhood of the grid should be iterated respecting a set of rules. However, how is it possible to obtain an Euclidean-like iteration process without using kernels in the GPU? That is, how is it possible to iterate through the neighbourhood in a discrete environment respecting the Euclidean distance? Kernels are slow. Furthermore, kernels of indefinite sizes are required in this case. We must also acknowledge the fact that grids are discrete spaces. This issue brought us to another solution, presented in Section 3.2.3, which is more similar to the Euclidean distance in terms of information while being outstandingly faster to compute when compared to Manhattan and Euclidean distances.

In 2 dimensions, the proposed distance metric resembles an octagon, while Chebyshev and Manhattan resemble a square. An octagon is closer to the circle obtained with the Euclidean distance. The 2D discrete solution for this distance was developed first. After months of calculations, the distance was successfully associated to a combination of Chebyshev and Minkowski distances, when generalized to p-dimensions. An extensive search throughout the literature was performed, which indicates that no work has ever proposed the combination of these distances, and most importantly, no work has ever associated the proposed Algorithm 6 (that iterates 2D neighbourhoods) to this combined distance.

1.2 Contributions

The main contributions of this thesis are:

1. ***Proposing and applying mathematical morphology techniques in machine learning.*** Mathematical morphology has never been properly explored in terms of classification. Works that use morphological operators to segment and classify images can be found in the literature (see Section 2.2.1). However, this is the entire extent to which these techniques are used. We propose a theoretical framework that works after the discretization/quantization of the space. Training instances are dilated according to a set of rules. An approach for processing p-dimensional and multi-class datasets is proposed.

   When it comes to clusterization, morphological reconstruction has never been used as a means of clustering data (see Section 2.3.1). Approaches that consist of using binary morphological operations have been proposed. Nevertheless, these algorithms

do not rely entirely on morphological operations. Instead, they look for cluster cores and use heuristics to categorize instances as belonging to one of the cluster cores. Unique differences between these approaches and the one provided in this work include the potential of removing noise without additional processing time burdens, different clusterization results and the possibility of controlling the maximal number of clusters. Furthermore, the approach also contains an intrinsic sense of the maximal amount of clusters that can be created.

2. ***Providing two classifiers that conform to item 1***. Two algorithmic proposals, namely Morphological Dilator Classifier (MDC) and Morphological k-Nearest Neighbours (MkNN) are presented. The first algorithm is primarily based on morphological dilations. The second approach is based on the classical k-NN in conjunction with mathematical morphology.

3. ***Providing one very efficient and unique clusterization algorithm that conforms to item 1***. The proposed approach, called k-Morphological Sets (k-MS), uses morphological reconstruction and gradual increases on the structuring element to form clusters. Morphological reconstructions are performed until the clusters reach an idempotent configuration. Next, the structuring element is incremented. Morphological reconstruction is performed again until reaching idempotence. This process is repeated until the desired number of clusters is obtained.

4. ***Proposing a novel and efficient distance metric***. The metric proposal (Section 3.2.3) is a combination of Minkowski and Chebyshev metrics. The most important contribution lays on its association with a very efficient 2D neighbourhood iteration. This distance is substantially faster than Euclidean and Manhattan in terms of 2D neighbourhood iterations.

1.3 Organization

The content of this thesis is organized as follows. Chapter 2 presents a literature review respecting the following order: (1) mathematical morphology and its applications, (2) classification algorithms, (3) clusterization algorithms, and (4) distance metrics.

Chapter 3 describes the proposed theory, methodology and implementation aspects. This chapter is also divided in two sections, which address the proposed classifiers (MDC and MkNN) and the clustering algorithm (k-MS), respectively. In a subsection of the

classifiers section (i.e., in Section 3.2.3), a novel metric is proposed and analysed. The metric is presented within this section as it was coined for the use with morphological classifiers. It defines the order in which the growth of the instances in the classification models occurs.

Next, Chapter 4 presents the experimental results considering all the implementations proposed in this thesis. In the classifiers case, we compare their efficiency to 14 algorithms evaluated on 8 UCI datasets. In this case, the proposed algorithms achieved reasonable results, outperforming or achieving the same accuracy of traditional classifiers in 5 cases.

Chapter 4.2 provides two experiments regarding the clusterization algorithm proposal. The first experiment demonstrates that there is in fact a major visual difference between clusterization algorithms that prioritize cluster density and shapes and clusterization approaches that do not. Later, another visual experiment is performed, where the running times obtained with state-of-the-art techniques are also compared. In this case, the proposal outperformed state-of-the-art methods in terms of processing times while providing very accurate results, sensitive to cluster shapes.

The final chapter presents the conclusions, benefits and disadvantages of the concepts herein. At last, avenues for future work are presented and discussed.

Chapter 2

Literature Review

The purpose of clustering and classification algorithms understand and extract value from large sets of structured and unstructured data. Both approaches are very important in science. Mathematical morphology, on the other hand, is traditionally used in image processing.

All these three concepts are reviewed throughout this chapter. The first section addresses some basic operations of mathematical morphology. The operations described herein are required to follow aspects of this thesis. Nevertheless, mathematical morphology does become deeper than that, using more complex operations and heuristics. It is possible to argue that the review contains the basic and intermediate operations of mathematical morphology. Watershed and morphological reconstructions, which are part of mathematical morphology, are sometimes considered complex operations.

Later, classification and clustering are addressed in two different subsections. Although both approaches belong to the machine learning nomenclature, classification and clustering are based on quite different premises and therefore are built upon quite different literatures, justifying this organization.

Classifiers train on data whose label is known *a priori*. Therefore, it is assumed *a priori* that unlabelled instances must be classified as one of the labels available in the training samples. Regression is also categorized as supervised learning. However, in this case, instances can be classified with labels that are not available in the training samples, as they are usually continuous. Nevertheless, labels of training samples are known in both cases.

Clusterization methods, on the other hand, learn from unlabelled data. This process can be better illustrated as grouping elements based on a similarity measurement. La-

belling data manually is a huge problem as datasets can be huge. Therefore, clusterization techniques are very important in this case, being able to infer and categorize data without relying on training labels.

Finally, the last subsection presents a brief review of distance metrics. Metrics are widely used in virtually all fields of exact sciences. In this thesis, they are specially important as they define the fashion of growth of the proposed classification algorithms.

2.1 Mathematical Morphology (MM)

Mathematical morphology [19, 58] is concerned with geometric structures present in sets. Analysis of image information is performed using set theory [26]. Structuring elements, which are used in most mathematical morphology operations, are defined by a mathematical set of elements that represents their pixels. Dilations and erosions are the two principles and fundamental operations in MM.

Definition A dilation of a binary image $A = \{a_1, a_2, ..., a_n\}$, containing n foreground pixels, by a structuring element B, is given by [98]:

$$dil^B(A) = \bigcup_{b \in B} A_b \tag{2.1}$$

where A_b represents the elements of A translated by b. In other words, dilations are defined by the union of $A + b$ for every element b in B. Each element represents the position of a pixel. Figure 2.1 illustrates a dilation of a binary image A by a cross-shaped structuring element $B = \{(0,0), (0,1), (0,-1), (1,0), (-1,0)\}$. The outcome is a less-noisy binary image, as several dark pixels were erased from the original image (it is evident on the character's face).

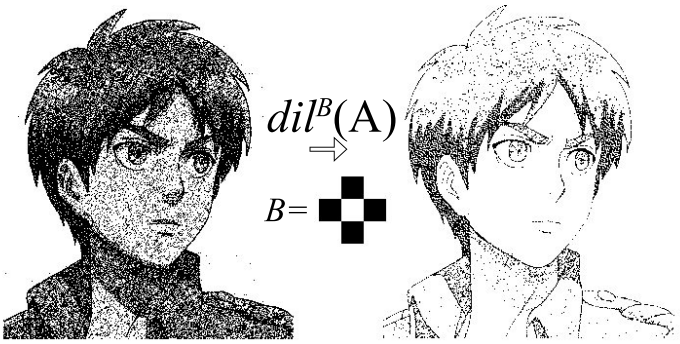


Figure 2.1: A binary image dilation using a cross-shaped structuring element.

Dilations set the intensity of any background pixels via the superposition of the structuring element on each of the foreground pixels of the binary image to the foreground value. In this specific case, since the spatial extent of the structuring element is fairly limited, it spreads the foreground pixels that already exist to their neighbouring pixels, thus filling the gaps of the original image in a symmetrical fashion.

Definition Erosion is the dual operation in relation to dilation and is defined as [98]:

$$er^B(A) = \bigcap_{b \in B} A_{-b} \tag{2.2}$$

Figure 2.2 shows the erosion of image A by the same cross-shaped structuring element B. In contrast, erosion erases pixels (in this representation, brighter pixels) respecting the structuring element. In this specific case, since the spatial extent of the structuring element is fairly limited, it erodes pixels near the original ones.

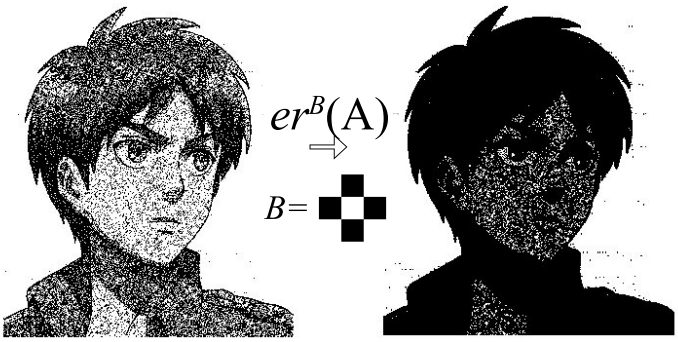


Figure 2.2: A binary image erosion using a cross-shaped structuring element.

Morphological operations can also be applied to grey level images [108]. In this case, the grey level value is accessed by evoking the image function. That is, $A(a)$ returns the grey level of image A at position a. This also holds for the structuring element, where $B(b)$ returns the grey level value of the structuring element at position b.

Definition The grey level dilation of a single element a in image A is given by:

$$dil^B(a) = sup_{b \in B}[A(a_b) + B(b)] \tag{2.3}$$

where *sup* represents the *supremum*, a_b represents the element a translated by b (e.g., given $a = \{0, 0\}$ and $b = \{-1, 0\}$, then $a_b = \{-1, 0\}$), $A(a_b)$ represents the grey value of the element a_b in A, and $B(b)$ the grey value of the element b in B.

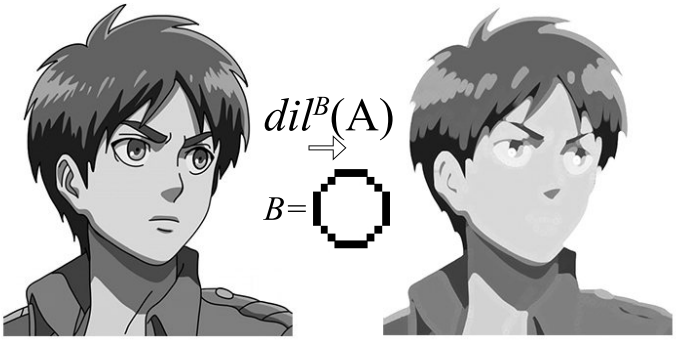


Figure 2.3: A dilation of a grey scale image using a circular structuring element (ring).

Definition The grey level dilation of the whole image A is given by:

$$dil^B(A) = \bigcup_{a \in A} dil^B(a) \tag{2.4}$$

Grey scale dilations translate the structuring element over A and compute the sum of the neighbouring pixel values (a's and b's) at each possible position (similar to a convolution). At every possible position a in A, the elements are summed and the *supremum* of these elements is placed at the iterated pixel of A. Figure 2.3 illustrates a grey dilation by a circular structuring element (also called ring).

Similarly, a grey level erosion adopts the *min* instead of the *sup*, as shown in Equation 2.5.

Definition Grey level image erosions of a single element a in image A is defined as:

$$er^B(a) = min_{b \in B}[A(a_b) + B(b)] \tag{2.5}$$

Definition The grey level erosion of the whole image A is given by:

$$er^B(A) = \bigcup_{a \in A} er^B(a) \tag{2.6}$$

When a grey level erosion is applied to a grey image using the same structuring element (ring), the result shown on the right side of Figure 2.4 is obtained.

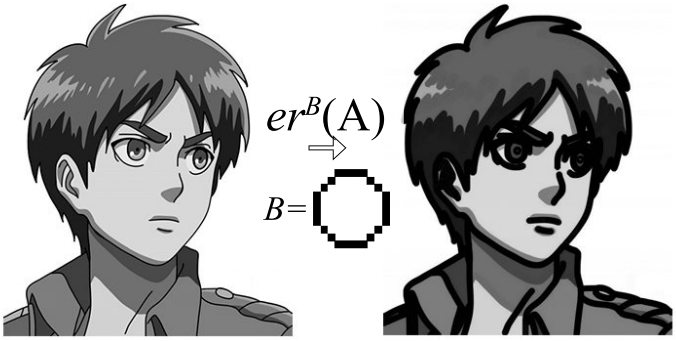


Figure 2.4: An erosion of a grey scale image using a circular structuring element (ring).

Opening and closing are operations built upon dilation and erosion and can be applied to grey level and binary images interchangeably [64].

Definition Opening is defined as first eroding and further dilating image A:

$$op^B(A) = dil^B(er^B(A)) \tag{2.7}$$

Figure 2.5 illustrates an opening using the same structuring element. It is noticeable that noise is promptly erased with this approach. Some of the highlights on the hair of the character are erased entirely without loosing much of the initial image information. Opening removes brighter components of the image, the size of the components to be removed varies according to the structuring element.

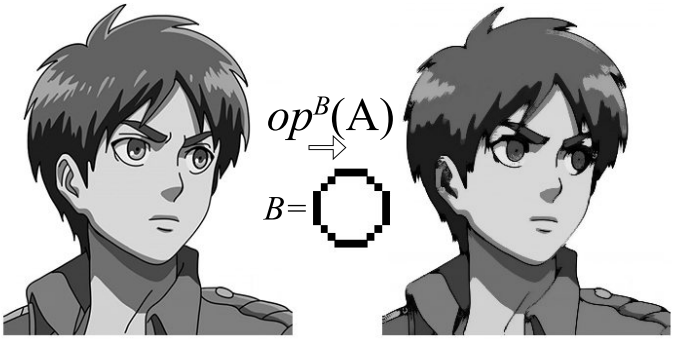


Figure 2.5: Opening of a grey scale image using a circular structuring element (ring).

Definition Closing is defined as dilating and subsequently eroding image A:

$$cl^B(A) = er^B(dil^B(A)) \tag{2.8}$$

Figure 2.6 illustrates closing using a ring as the structuring element. It is noticeable that all the black outline is erased while trying to maintain the original shapes of the input image. Closing, as opposed to the previous operation, removes darker components of the image.

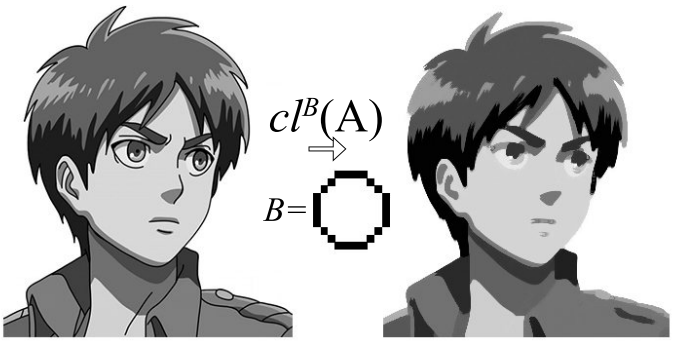


Figure 2.6: Closing of a grey scale image using a circular structuring element (ring).

White and black top-hat operations use opening and closing, respectively. These operations subtract the result of the opening or closing from the original image. It is a very interesting way to remove noise from images and is used as such throughout the literature. Approaches that use this operation to segment objects/edge detection are also widely present in the literature [6, 16].

Definition White top-hat (or simply top-hat) is defined as subtracting an image from its opening [24]:

$$wth^B(A) = A - op^B(A) \tag{2.9}$$

Figure 2.7 illustrates a white top-hat transform. The result displays noise that is present in the original image. This operation returns the small components (varies according to the size of the structuring element) that are brighter than their surroundings.

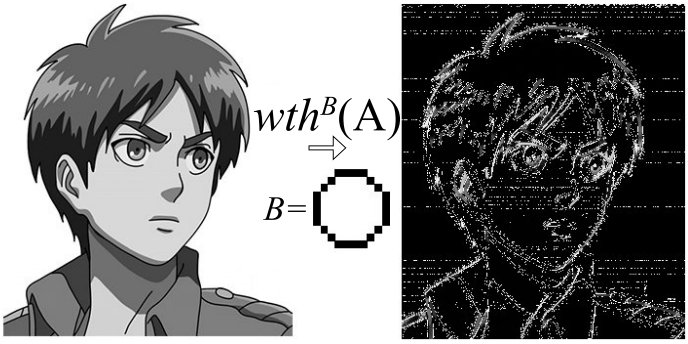


Figure 2.7: A white top-hat transform of a grey scale image using a circular structuring element (ring).

The black top-hat operation (or bottom-hat transform) is the dual operation of the white top-hat. Figure 2.8 shows a black top-hat transform. The outcome represents small components (shown in black, and varies according to the size of the structuring element) that are darker than their surroundings.

Definition Black top-hat (or bottom-hat) is defined as subtracting the closing of an image from its original [24]:

$$bth^B(A) = cl^B(A) - A \tag{2.10}$$

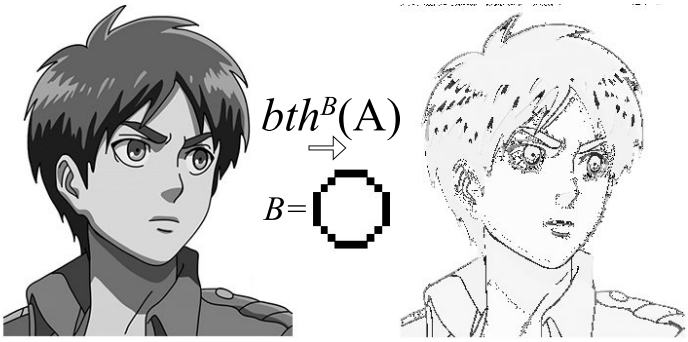


Figure 2.8: A black top-hat transform of a grey scale image using a circular structuring element (ring).

Geodesic operations use a binary mask M that limits the range of the operation.

Definition Geodesic dilations are defined as:

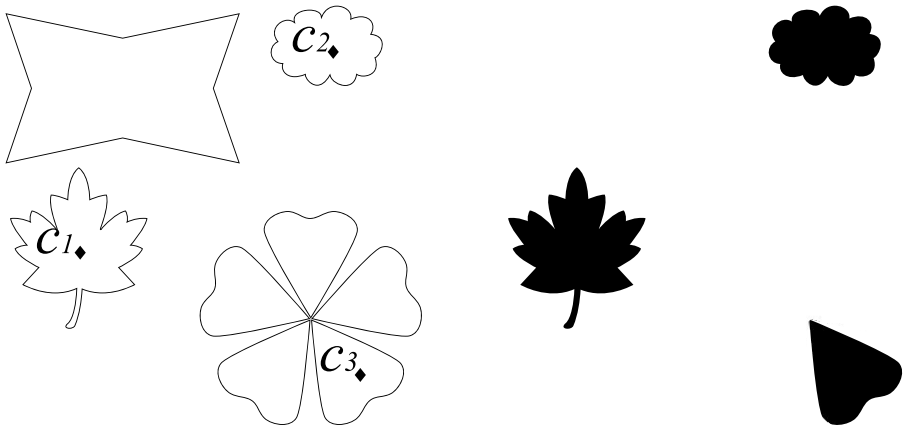
$$dil^B(A) \cap M \tag{2.11}$$

Definition Similarly, geodesic erosions are given by:

$$er^B(A) \cap M \tag{2.12}$$

Morphological reconstructions adopt geodesic dilations [68]. That is, the reconstruction dilates images using a mask that limits the range of operation. The mask can be described as a simple binary image. Given a single point in the image and a mask, the image is dilated until it reaches idempotence. Idempotence is defined as $f(A) = A$, where f in this case is a morphological operation and A is an arbitrary image [33].

Let the lines illustrated in Figure 2.9-(a) represent the mask and the points c_1, c_2 and c_3 illustrate seed points to be dilated. The resulting image after idempotence is shown in Figure 2.9-(b).



(a) Mask and points to be dilated (b) After the reconstruction

Figure 2.9: Morphological reconstruction (after idempotence).

Definition Morphological reconstruction is defined as $R_M^B(A)$ in Equation 2.13, where M represents the mask.

$$R_M^B(A) = h_Q \tag{2.13}$$

subject to

$$\begin{aligned} h_{q+1} &= dil^B(h_q) \cap M \\ &\vdots \\ h_1 &= A \end{aligned} \tag{2.14}$$

where q stands for every iteration until idempotence ($q \in \{1, ..., Q\}$) at iteration Q, i.e., $h_Q = h_{Q-1}$.

The binary morphological reconstruction is not a clustering algorithm by itself. It spreads pixels in the image limited by mask M and the process is terminated when idempotence is reached. The outcome is a binary image, such as the one shown in Figure 2.9-(b).

When an appropriate grey-level structuring element and grey level images are used, while also indexing each pixel with a unique value and still using the same mask M, clusters start to form in the outcome image, where each object of Figure 2.9-(b) would be in a different grey color, associated to a unique label. The main issue with grey scale morphological reconstructions is that there is no k variable. That is, it is not possible to limit the amount of clusters to be created using a simple morphological reconstruction. Furthermore, once the images reach idempotence, it is not possible to merge clusters with the original formulation.

Watershed is a more complex morphological operation. It segments grey level images based on local minimas. Grey level images can be seen as height-maps. Watersheds are usually constructed by flooding [62], though other approaches exist. Let us suppose filling valleys of a mountain with water simultaneously. The moment in which the flooding or the water of two or more valleys intersect is defined as a watershed ridge. Figure 2.10 illustrates this process.

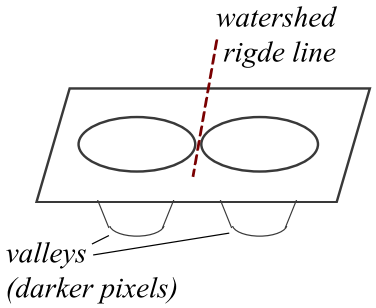
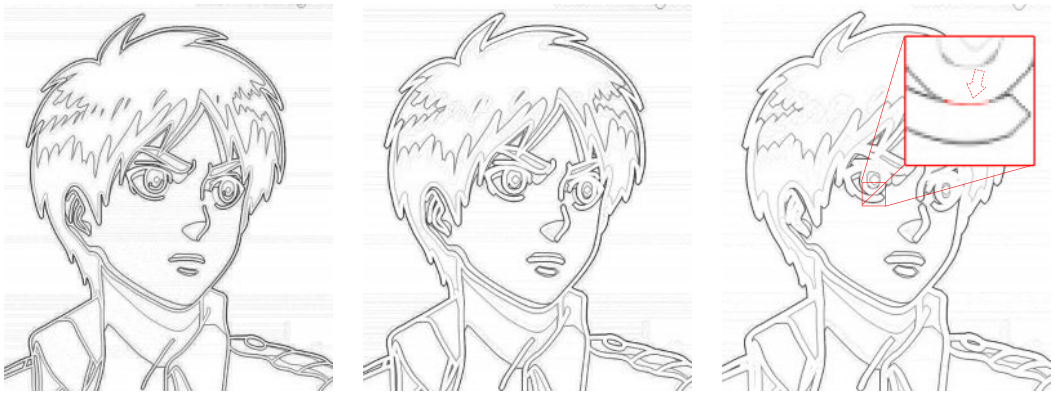


Figure 2.10: Watershed ridge (intersection of two local minimas).

Watershed ridges can be calculated using erosions. The original image is subtracted by its eroded version, which represents the flooding of the darkest grey values. Figure 2.11 illustrates this operation being applied in sequence. In Figure 2.11-(c), an intersection of two ridges occurs, which is a watershed ridge (highlighted in red).



(a) Original image subtracted by its eroded version ($A - er(A)$).

(b) Eroded version subtracted by the eroded version of the eroded version ($er(A) - er(er(A))$).

(c) Two times eroded version subtracted by the three times eroded version ($er(er(A)) - er(er(er(A)))$).

Figure 2.11: A morphological approach to watersheds.

Figure 2.12 demonstrates a possible watershed outcome. In the first case 2.12-(a), the watershed is computed from the original input image. In 2.12-(b), a Gaussian blur is applied before calculating the watershed, which alters significantly the produced segmentation. The original image has a high incidence of noise, which is clear due to the oversegmentation shown in 2.12-(a), and is mitigated when the Gaussian blur is applied (it softens the noise as it is a mean filter).

(a) Watershed applied to the input image.

(b) Watershed after a Gaussian blur.

Figure 2.12: Results of the watershed algorithm.

2.1.1 Mathematical Morphology Applications

In general, mathematical morphology is employed in the literature for geometric feature filtering, e.g., to segment structures [94, 110], extracting and enhancing blood vessels [25, 2], etc. Liu et al. [54] use grey level mathematical morphology operations and a genetic algorithm to create a thresholding method for images. Furthermore, it is also used to remove noise [119, 120]. The noise removal potential is evidenced in Figures 2.1-2.7 [117].

The closest that mathematical morphology comes to classification in the present literature is by providing image descriptors for further classification [93, 55]. In contrast, clusterization methods that use morphological operations already permeate the literature. However, as properly addressed in Section 2.3.1, these clusterization methods: (1) rely on binary operations, (2) are coined for images, and (3) neither use morphological reconstruction nor a k/number-of-clusters control parameter as input. The power of mathematical morphology has yet to be fully exploited in the context of machine learning, which is the main contribution of this thesis.

2.2 Classification

Given a training dataset $X = \{\mathrm{x}_1, ..., \mathrm{x}_n\}$ containing instances such that $\mathrm{x} \in \mathbb{R}^p$, p being the number of attributes of the instances, with corresponding labels $c(\mathrm{x}) \in \{1, ..., L\}$, the classification problem consists of assigning a label $l \in \{1, ..., L\}$ to unlabelled instances y, given that X can assist the process. In this formulation we assume that each categorical attribute, if present, is associated to natural numbers.

Several types of classifiers permeate machine learning. Lazy algorithms [100], for instance, generate classification models in real time. That is, every time an instance is to be classified, the algorithm generates a model as a requirement to classify the instance. Their main weakness stems from that fact, i.e., generating new models when an unlabelled instance is encountered, which is inefficient in terms of processing times. However, this results into a significant strength, i.e., there is no requirement to store a model in memory.

k-NN is one of the most popular lazy algorithms and probably the simplest one. It requires the k nearest neighbours of an unlabelled instance to be examined in order to determine its class. The predominant class among the k neighbours is the chosen class. k-NN is a robust algorithm in terms of accuracy, yet it does not deal very well with noise in

some specific cases and for some values of k. One such occasion is when a small number of instances of one class is surrounded by several instances of another class. A simple k-NN with $k = 3$ is depicted in Figure 2.13, where the unlabelled instance y is checked against the 3 nearest labelled instances and is classified as the most frequently encountered class which, in this case, is the square class.

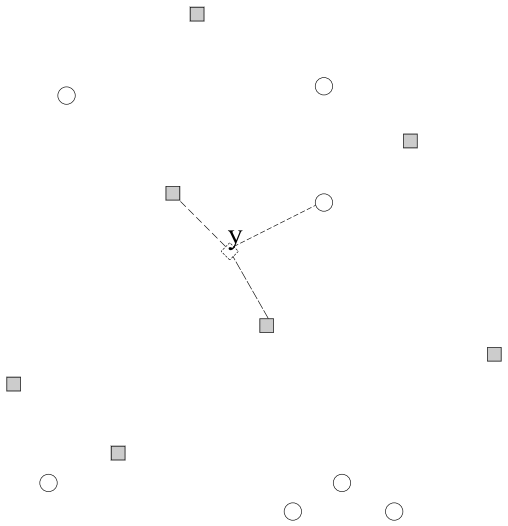


Figure 2.13: Example of a k-NN algorithm in a 2D plane.

Decision tree based classification relies on decision tree models. Essentially, these algorithms generate one or more trees that, in turn, contain several nodes. Each node split is usually associated to a formulated Boolean criterion that is dependent on one or more attributes of the dataset. For instance, $a_i > s$, where a_i is the attribute at index i and s is a constant. The split governs the flow of the decision, and the leaves of the trees represent the various classes. A popular decision tree algorithm is C4.5 [76, 103] (J48 in Weka [36]). Random Forests [10] is also a frequently employed algorithm that generates several decision trees, which are then combined, composing an ensemble learning method [67]. Figure 2.14 illustrates a possible decision tree based classification model.

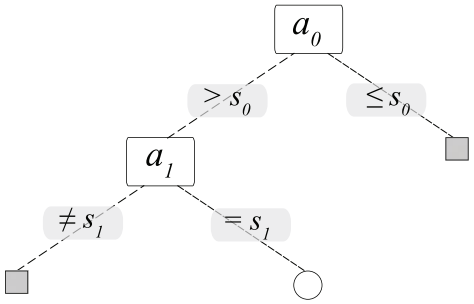


Figure 2.14: A decision tree model with two nodes (two split criterions).

Another type of classifiers is neural networks. Neural networks are often categorized as function-based classifiers. They associate input patterns to output patterns by using the representation of intermediate layers, activation functions and weights that connect the various nodes, mimicking the organization of the brain. Two popular neural networks are the Radial Basis Function Network (RBFNetwork) [11, 12] and the Multilayer Perceptron [38, 91]. Figure 2.15 shows a feedforward neural network with n input units, m hidden units and two outputs. The outputs of the hidden units are the weighted sums of the input attributes after they pass through the activation functions of these units. Finally, the outputs of this neural network are calculated as the weighted sums of the input from the hidden units after they pass through the activation function of the output units.

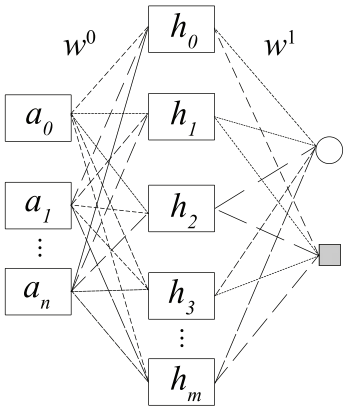


Figure 2.15: A simple neural network with one hidden layer consisting of m units, n inputs and l outputs (in this case, a binary output).

Another type of function-based classifiers is Support Vector Machines (SVM). In summary, SVM traces hyperplanes that segregate the data, and uses the created clusters or segregations to predict classes. In its basic form, when no kernel is used, SVM works in a straightforward fashion. Figure 2.16 illustrates a possible solution, shown by the dashed line, which separates the data optimally. Optimality in this case refers to higher accuracy and margin, i.e., gap size g. Since this problem is 2-dimensional, the hyper-plane is a simple line. Algorithms based on SVM such as Sequential Minimal Optimization (SMO) [73] also bear the same optimization objective. In Figure 2.16, the training instances are separated by the dashed line such that if an unlabelled instance falls on the left/right side of the line, it would be classified as a square/circle, respectively.

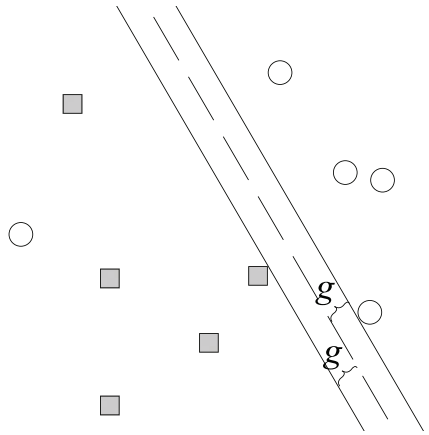


Figure 2.16: A possible 2D SVM model.

Some of the previously described algorithms can be thought of as probabilistic models. That is, some decision tree algorithms, for instance, output a probability on the correctness of classification of a certain node, or the respective path on the tree that a given instance followed. These probabilities can be computed in various ways and therefore these algorithms could be considered probabilistic. However, some classifiers are based on conditional probabilities, more specifically, on Bayes Theorem [49]. These are explicitly categorized as probabilistic. Here, we refer to them as Bayes-based algorithms, with the popular algorithms being the Naive Bayes [101, 78] and Bayes Net [59, 97]. In Bayes theorem, $P(E_i)$ denotes the probability of a given event E_i being true. The event could be, for instance, the probability of an attribute being equal, greater, lower or different than a certain value. $P(E_a|E_b)$ denotes the conditional probability, which is the probability of E_a being true given that E_b is true. A simple Bayes net is shown in Figure 2.17. The joint probabilities at the right of the figure indicate the probability of the analysed instance belonging to a certain class, given the described events. Exclamation marks indicate negation.

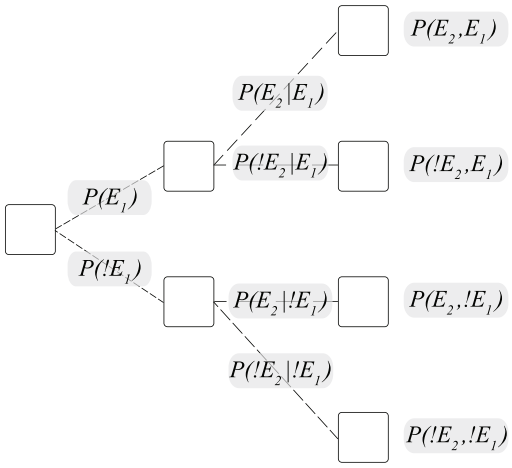


Figure 2.17: A simple Bayes network.

Lazy, decision tree, function, and Bayes classifiers are probably the most commonly used classifers in the state-of-the-art. However, another category of classifiers that often produces very good results yet is often disregarded, are rule-based classifiers. Decision Table [46] and Conjuctive Rule [36] are two such examples. A decision table can be extraordinarily simple, for instance, a default rule mapping to the majority class. The generated table can also be relatively easy to understand, as simple rules can be intentionally generated. The process involves heuristical search of subsets of features for rule finding, as well as cross-validation [46].

Figure 2.18 shows a decision table model. The first variables r_0, r_1 and r_2 represent rules based on the instances used to train the classifier. In the second column, for instance, if rules r_0 and r_1 are true, then the instance is said to be a circle. In the fifth column, on the other hand, if an unlabelled instance has rules r_0 and r_1 equal to false and r_2 as true, then it is classified as a square. The rule itself can be a simple comparison split of an attribute (e.g., if an attribute value is higher than a constant), as in decision tree models, or more complex associations.

r_0	T	T	F	F		T		F
r_1	T	F	T	F			T	F
r_2				T	T			
○	✔	✔			✔		✔	
□			✔	✔		✔		✔

Figure 2.18: A decision table that predicts the class of unlabelled instances based on the rules r_0, r_1 and r_2. If r_0 and r_1 are true, the instance is probably a circle.

2.2.1 Mathematical Morphology in Classification

To the best of our knowledge, mathematical morphology has not yet been properly exploited when it comes to classification. Pina et al. [71] use a combination of BMCAs (see Section 2.3.1 for further details) and watershed, creating a classification method for satellite images. The algorithm creates cluster cores for each class and uses these cores to decide to where each unlabelled instance belongs. The authors use a watershed approach, which produces segmentations for the whole image. Their method is better defined as an image segmentation approach that uses supervised learning.

Tuia et al. [106] use features provided by morphological opening and closing to classify land use in satellite images. In this case, as opposed to the previously described work, SVM was the employed classification algorithm. Mlynarczuk [64] uses watershed to classify images of rocks, again as a descriptor instead of a classification algorithm itself. Shattuck et al. [95] use dilation, erosion and closing to perform brain segmentation/classification. In summary, several works considered using mathematical morphology for image classification. However, none of them use mathematical morphology as a means of constructing proper classification models that are able to deal with p-dimensional data.

Having presented an overview of various classification models, we will shift our attention to the work of Bien et al. [8], which shares some similarities to the proposed classification methods of this thesis (without using mathematical morphology). A training dataset $X = \{\mathrm{x}_1, ..., \mathrm{x}_n\} \subset \mathbb{R}^p$ is considered, where p represents its dimension, along with prototype sets $P_l \subseteq X$ for each class l, which are used to classify unlabelled instances. At the heart of their proposal is the premise that the prototypes of class l should consist of points that are close to many training examples of class l and far from training examples of other classes. Given prototypes $P_1, ..., P_L$, an unlabelled instance $\mathrm{x} \subset \mathbb{R}^p$ is classified according to which of the P_l sets contains the nearest prototype, based on some distance metric d:

$$\hat{c}(\mathrm{x}) = \arg \min_{l} \min_{z \in P_l} d(\mathrm{x}, z) \tag{2.15}$$

Despite the conceptual similarities to this work, the proposed approach bears significant differences. The first difference is that our approach considers the shape information of the classes. Although this is also partially true for the Bien et al. approach, our approach considers this explicitly and enables the use of operations that can pre-process this data if necessary. For instance, applying erosion, opening or closing operations to remove

noise from the dataset.

Besides, the theory proposed in this thesis does not necessarily lay on the same premise that each prototype of class l consists of points that are close to most training examples of class l and far from training examples of other classes, although this may be true for most datasets in practice. However, in occasions such as when one class is a subtle translation of the other class, as shown in Figure 2.19, their prototype approach does not perform reasonably well. A similar situation is also found when, for instance, both classes follow a fractal or a pseudo-random noise function. The positions of the points belonging to each class can be exactly predicted if the function is known *a priori*. However, if this information is not known, the premise considered by Bien et al. does not hold.

Figure 2.19: Class 1 instances are shown in black and class 2 in grey. Both classes are the exact translation of the other class.

We overcome this limitation by avoiding the concept of prototypes. Instead, we exploit the use of morphology-preserving transformations such as the ones offered by MM to expand sets of pattern points corresponding to classes in discretized feature or classification space. Using this p-dimensional shape preserving approach [42], the development of classification models, capable of capturing the topological features of distinct classes, is enabled. Such an example in 2 dimensions is shown in Figure 2.19, which can be solved with zero classification errors, unlike the Bien et al. approach. At last, Bien et al. also highlight that the main strengths of their methodology lie in the ease of understanding why a given prediction has been made an alternative to (possibly high-accuracy) "black box" methods. This statement also holds for our methodology.

2.3 Clustering

In contrast to classification, the literature on clustering contains some works that use mathematical morphology techniques. Still, these techniques are not properly exploited.

Given a dataset $\mathrm{X} = \{\mathrm{x}_1, ..., \mathrm{x}_L\}$, containing L instances, where $\mathrm{x}_l \in \mathbb{R}^n$, $0 < l < L$, the common clustering problem consists of partitioning X into k disjoint subsets or clusters $(C_1, ..., C_k)$ such that a cluster criterion is optimized. The most widely used clustering criterion is the sum of the squared Euclidean distances between each instance x_l and its centroid u_m (cluster center) of the subset that contains x_l [51]. This clustering error is defined in Equation 2.16:

$$E(C_1, ..., C_k) = \sum_{l=1}^{L} \sum_{m=1}^{k} I(\mathrm{x}_l \in C_m)||\mathrm{x}_l - u_m|| \tag{2.16}$$

where $I(Y) = 1$ if Y is true or $I(Y) = 0$, otherwise [51], and $||\ ||$ represents the distance norm.

In terms of applications, clustering techniques are used in a broad range of areas such as in (1) knowledge discovery, e.g., to separate instances that have common characteristics in different categories [29, 63, 32], in (2) image processing and analysis, to categorize types of tumours, diseases and abnormalities [96, 34], in (3) pattern recognition and image segmentation, to infer the content of images [102, 27], in (4) adaptive controllers for games [105], where the controls are adapted in real time according to the input of the user, in (5) robotics [45] for navigation, and in many other instances.

Clustering algorithms usually conform to one or more of the following categories [113]:

1. **Centroid-based clustering**: Clusters are created respecting a similarity measure. There is no hierarchy of clusters. Clusters can be adapted in real time. The problem consists of finding k cluster centers, further assigning points to the nearest cluster center. k-Means algorithm falls in this category.

2. **Hierarchical**: As the name suggests, algorithms that fall in this category seek to build a hierarchy of clusters. Distance measures can be used with hierarchical clustering as well. Strategies usually fall into two categories:

   (a) **Agglomerative**: Individual, smaller clusters are created initially. Pairs of clusters are merged as time advances, moving up on the hierarchy.

   (b) **Divisive**: Huge clusters are created at first (usually just a single big cluster). In this case, the bigger clusters are divided in time until the desired number of clusters is obtained.

3. **Statistical clustering**: Also called distribution-based clustering, these algorithms rely on mathematical models or statistics to generate clusters. A simple approach is generating Gaussian distributions over the data (initialized randomly), whose parameters are adapted iteratively over time. Algorithms such as EM (Expectation Maximization) fall in this category.

4. **Density-based clustering**: Clusters are usually defined as areas of higher density. Points that fall in sparse areas are usually considered noise.

k-Means is arguably the most popular clustering algorithm. Although robust and simple, it does not consider morphological aspects. In other words, k-Means is not able to exploit shape and density information. On the contrary, the proposed clustering algorithm overcomes these limitations by using morphological reconstructions, which are sensitive to cluster density and to the morphology itself. In this research, we propose and extend a morphological reconstruction algorithm to include the input parameter k.

Kapyris et al. [44] proposed a clustering algorithm called Chameleon that is capable of indirectly detecting morphology in datasets. However, their algorithm is not based on mathematical morphology itself. Instead, it consists of two phases. In the first phase it uses graph-partitioning to cluster the dataset in relatively small sub-clusters, while in the second phase, it uses an algorithm to find the genuine clusters by repeatedly combining these sub-clusters. Chameleon is categorized as a hierarchical clustering algorithm, which seeks to build a hierarchy of clusters. It can also be categorized as agglomerative instead of divisive because clusters start small and are aggregated to form bigger clusters as time advances, rather than the opposite, i.e., starting with large clusters and dividing them in time.

It is clear that Chameleon is based on graph operations, which are very different in relation to mathematical morphology. Although their algorithm is able to indirectly detect morphology, it lacks the richness of configurations that are enabled through mathematical morphology. In fact, with the clustering algorithm proposed in this thesis, we are not only able to change the k variable; the structuring element can be replaced as well, which alters the fashion in which the clusters propagate. Besides, it is possible to perform pre-processings such as dilations and erosions or combinations of them before and after clustering images.

Liu et al. [52] proposed a clustering algorithm called TRICLUST, which is based on the Delaunay triangulation. Furthermore, Liu et al. [53] proposed a clustering algorithm

that is similar to previously described approaches. The authors evaluate how well cluster prototypes are separated and form clusters using a separation measure.

Finally, Yousri et al. [114] proposed a clustering algorithm called Mitosis. This approach also bears similarities to previous work. In this case, Mitosis uses distance relatedness patterns as a measure to identifying clusters of different densities. The authors point out some issues of the Chameleon algorithm such as the slow speed of the algorithm and the difficulty in tuning its parameters. These four works are the most similar to the proposed clustering algorithm. It is important to highlight that they do not use concepts from mathematical morphology.

2.3.1 Mathematical Morphology in Clusterization

Postaire et al. [74] proposed the first approach to mathematical morphology clusterization. Although not defined by the authors in their original work, these types of algorithms were later termed Binary Morphology Clustering Algorithms (BMCAs) [69, 56]. Figure 2.20 illustrates how the process of finding cluster "cores" work.

(a) Original data.

(b) Opened twice and closed twice.

(c) Opened twice and closed four times.

(d) Opened twice and closed six times.

Figure 2.20: Cluster core extraction using BMCAs.

Opening and closing operations are successively applied to data until reaching connected core clusters, as shown in Figure 2.20-(d). Postaire et al. [74] defines the cluster as a well-connected subset in the data space, which can virtually be of any shape and size. Once the cluster cores are generated, they are considered to be prototypes. The remaining data points are assigned to their respective cluster by means of the nearest neighbour classification rule.

Pedrino et al. [69] introduced an evolutionary algorithm to be used in conjunction with the original BMCA approach. The idea is fundamentally the same, the input data is discretized/quantized and opening/closing operations are used to locate cluster cores.

However, as a proper structuring element is fairly difficult to guess beforehand, the authors improved on this aspect using an optimization algorithm.

Luo et al. [56] advocates for the use of brute force when it comes to selecting the structuring element assuming that no *a priori* knowledge is given. Their work is very similar to Pedrino et al. [69], the main difference lays on the fact that the authors use morphological operations to concatenate instances to the core instead of the k-nearest approach. All these BMCA algorithms do not accept a k variable, i.e., the number of clusters is found automatically.

BMCAs [74, 69, 56, 15] rely entirely on binary operations, as opposed to the algorithm proposed in this work, which works with grey level dilations/image reconstruction. A substantial and fundamental difference is that we do not compute nor locate cluster cores. In our case, elements are promptly assigned to a cluster index as the processing advances. This introduces a noise-classification pattern with no extra processing burden, as noise is segmented in small independent clusters. Furthermore, the clusterization results are very different while also providing a means of limiting the amount of clusters.

Some algorithms focus on image segmentation/clusterization [7, 30]. These algorithms create clusters based on the image grey level information. Among other operations, the morphological watershed operation is used for this matter. The clustering algorithm proposed in this work uses grey level information just as a means of indexing each data point in the grid/quantized space. Therefore, the clusterized outcome is not tuned towards image grey level content. In fact, it is directed towards usual clustering information while being capable of exploiting density and shapes.

2.4 Metrics

Finally, as a novel distance metric is proposed in this thesis, a brief literature review on distance metrics is performed in this section. A metric space is a pair (χ, d) where χ is a set and d is a mapping from $\chi \times \chi$ into $\mathbb{R}$, which satisfies the following conditions. Let χ be an arbitrary nonempty set and d a real-valued function on the Cartesian product $\chi \times \chi$:

$$d : \chi \times \chi \to \mathbb{R} \tag{2.17}$$

and consider the following four properties, which hold for arbitrary points $x, y, z \in \chi$.

1. $d(x, y) \geq 0$ (symmetry)
2. $d(x, y) = 0$ iff $x = y$ (non-negativeness)
3. $d(x, y) = d(y, x)$ (positiveness)
4. $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)

A real-valued function d on $\chi \times \chi$ that satisfies all four properties is a metric in χ, and the properties themselves are called the metric axioms. A set χ equipped with a metric d on $\chi \times \chi$ is a metric space, also denoted by the pair (χ, d). When d satisfies all but the third property, it is called pseudometric. A distance function is any real-valued function d on $\chi \times \chi$ that satisfies property 1 and 2. Similarly, when the fourth property is not satisfied, it is called semimetric [18].

Example Let χ be a non-empty set and d:

$$d(x, y) = \begin{cases} 1, & \text{if } x = y \\ 0, & \text{if } x \neq y \end{cases} \tag{2.18}$$

(χ, d) is a discrete metric space.

The Euclidean distance [109] is the most commonly used metric when it comes to k-NN and other classification algorithms [111]. This may be associated to the fact that we are immersed in an Euclidean space, and therefore the Euclidean distance naturally comes to mind. However, other distances outperform the Euclidean in several occasions, which may be unintuitive at first glance [18].

Euclides once stated that the shortest distance between two points is a line. This assertion popularized the Pythagorean metric as Euclidean distance, although derived from the Pythagorean Theorem [14]. The Euclidean distance $d_2(x, y)$ between points x and y, where $x, y \in \mathbb{R}^n$, is given by Equation 2.19, where n represents the number of dimensions in this subsection only.

Definition Euclidean distance:

$$d_2(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.19}$$

The Manhattan distance [35] was proposed by Hermann Minkowski in the late 19^{th} century [14] and is defined as the sum of the absolute differences of the Cartesian coordinates, as shown in Equation 2.20.

Definition Manhattan distance:

$$d_1(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{2.20}$$

Later, Minkowski [43] included an exponent p in its formulation. Minkowski distance [3], coined after him, is a generalization of both Euclidean ($p = 2$) and Manhattan ($p = 1$) distances. The metric conditions are satisfied as long as p is equal or greater than 1. $p < 1$ violates the triangle inequality (fourth condition).

Definition Minkowski distance:

$$d_p(x, y) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p} \tag{2.21}$$

When p reaches positive infinity, the Chebyshev distance [14] is obtained, as shown in Equation 2.22.

Definition Chebyshev distance:

$$d_\infty(x, y) = \max_{i=1}^{n} |x_i - y_i| \tag{2.22}$$

These 4 distances metrics, Euclidean, Manhattan, Minkowski and Chebyshev, are the mostly commonly known and consequently the mostly commonly applied metrics in general. However, the Squared Euclidean distance and the Canberra distance are also considered in this thesis.

The Squared Euclidean d_{SD} is also known as the Sum of Squared Difference. This is the fundamental metric in least squares problems and linear algebra, and is shown in Equation 2.23.

Definition Squared Euclidean distance:

$$d_{SD}(x, y) = \sum_{i=1}^{n} (x_i - y_i)^2 \tag{2.23}$$

Finally, the Canberra distance is a weighted version of the Manhattan distance initially proposed by Lance et al. [48] in the '60s, and is shown in Equation 2.24.

Definition Canberra distance:

$$d_{CAD}(x, y) = \sum_{i=1}^{n} \frac{|x_i - y_i|}{|x_i| + |y_i|} \tag{2.24}$$

Cha [14] provides a complete survey on similarity measures. A total of 45 variations of similarity measures are addressed in his work. A general statistical analysis is performed, comparing the measures pairwise. No machine learning experiments were presented.

2.5 Summary

This chapter performed an extensive literature review in classification and clusterization including their relationship with mathematical morphology. Furthermore, a brief review on metrics was included, as a new distance metric is one of the contributions of this thesis.

As previously stated, mathematical morphology is not sufficiently employed neither in classification nor clusterization. Clusterization, however, does contain a small associated literature that uses mathematical morphology. Presumably, this is due to clustering being a major problem in visual computing, and the connection between these concepts is much more straight-forward considering images than classification is.

The existing literature on clustering can be summarized as BMCAs, where cluster cores are found using binary morphological operations. The clustering proposal of this thesis uses grey level image operators and a novel way of selecting the desired amount of clusters *a priori*. The BMCA classification proposal, on the other hand, uses binary operations. However, no work has ever attempted to do this in classification. A single work in the literature proposes an image segmentation based on classification, for images only. In addition, morphology is commonly used as a descriptor or as a feature of the image space, but not as a classification algorithm itself.

In terms of clusterization, other works in the literature propose different methods, based on fundamentals other than mathematical morphology, that are sensitive to cluster density and shapes. The proposed approach is also sensitive to these aspects, and at the same time is based on different premises and is more efficient than these related proposals in terms of processing times. Comparisons regarding these algorithms can be found in Chapter 4. The following chapter addresses the proposed methodology in detail.

Chapter 3

Morphological Methodology

This chapter describes the proposed methods in classification and clusterization, respectively. Before introducing each implementation, we provide their theoretical framework. In the following Section 3.1, we provide some definitions [37] that belong to set theory and are used throughout this chapter.

Section 3.2 addresses the classification framework. Two proposed implementations are addressed at first (MkNN and MDC). Later, the proposed metric is presented and analysed, as it is necessary for the classification process and for the complexity analysis. In what follows, a complexity analysis of both implementations is performed. An approach for the combination of 2-dimensional classification models is provided along with approaches for reducing the storage requirements of the classification models.

Section 3.3 presents the proposed clusterization algorithm (k-MS). A complexity analysis of the clustering approach is provided in sequence. Finally, parallelization aspects considering the GPU architecture and memory aspects are addressed and discussed.

3.1 Definitions

Definition *Set:* A set is defined as a collection of distinct elements. For instance, numbers 1, 2 and 3, considered collectively form a set $\{1, 2, 3\}$.

Definition *Subset:* A set A is a subset of B if A is contained in B. That is, all elements of A are also elements of B. Subsets are denoted as $A \subset B$. $A \subseteq B$ indicates that A can also be equal to B.

Definition *Multiset:* A multiset is a set that aggregates repeated elements. $\{1, 1, 2, 3\}$

and $\{1, 2, 3\}$ relate to the same set but are different multisets.

Definition *Partition:* A partition of a set is a grouping of non-empty subsets. For instance, $\{1, 2, 3\}$ has five partitions: $\{\{1\}, \{2\}, \{3\}\}$, $\{\{1, 2\}, \{3\}\}$, $\{\{1, 3\}, 2\}$, $\{\{1\}, \{2, 3\}\}$, $\{\{1, 2, 3\}\}$.

Definition *Element:* An element is a unique member of a set/multiset. As an example, the multiset $\{1, 1, 2, 3\}$ contains four elements (its cardinality is equal to 4): 1, 1, 2 and 3, respectively.

3.2 Classification

The essence of the approach for supervised learning is to expand pattern sets such that the expansion covers the entire feature space, while preserving shapes and density of the original clusters of labelled instances. We commence a process where subsets are expanded using set operations from their original state $i = 0$, to their final state, $i = I_l$, for each class l. For this, we focus on data-driven classification functions, based on the p-dimensional shape of the original class clusters. The expansion may be terminated in a specific direction if a suitable termination criterion is reached. This termination criterion can be, for instance, the classification error $P_{err} > t_{err}$, where t_{err} is a predefined misclassification threshold.

We consider a class of labelled patterns represented by a number of disjoint clusters as a multiset. As previously addressed, a multiset is a generalization of the sets concept which allows the presence of repeated instances. Thus, given a training dataset/multiset $X = \{\mathrm{x}_1, ..., \mathrm{x}_n\}$ containing instances such that $\mathrm{x} \in \mathbb{S}$, where $\mathbb{S} \subset \mathbb{R}^p$, p being the number of attributes of the instances, with corresponding labels $c(\mathrm{x}) \in \{1, ..., L\}$, an unlabelled instance y is assigned a label $l \in \{1, ..., L\}$, such that:

$$\hat{c}(\mathrm{y}) = l : \mathrm{y} \in A_l^{I_l} \tag{3.1}$$

subject to

$$\bigcup_{l=1}^{L} A_l^{I_l} = \mathbb{S} \tag{3.2}$$

where I_l corresponds to the final iteration for class l and $A_l^{I_l}$ is the expanded set partition of instances belonging to class l, which consists of J_l disjoint subsets $a_{l,j}^{I_l}$ such that:

$$A_l^{I_l} = \bigcup_{j=1}^{J_l} a_{l,j}^{I_l} \tag{3.3}$$

In what follows, we describe the methodology for obtaining the expanded set partition $A_l^{I_l}$. Let us assume a set of pattern points A_l^0 belonging to class l, with index 0 referring to the original set partition of training samples:

$$A_l^0 = \{\mathrm{x} \in X \ : \ c(\mathrm{x}) = l\} \tag{3.4}$$

The original set partition of samples belonging to class l, A_l^0, is represented by the union of the disjoint subsets $a_{l,j}^0$. An example of this in two dimensions is shown in Figure 3.1. Here, the class set partition A_l^0 is the union of $\{a_{l,1}^0, a_{l,2}^0, a_{l,3}^0, a_{l,4}^0\}$.

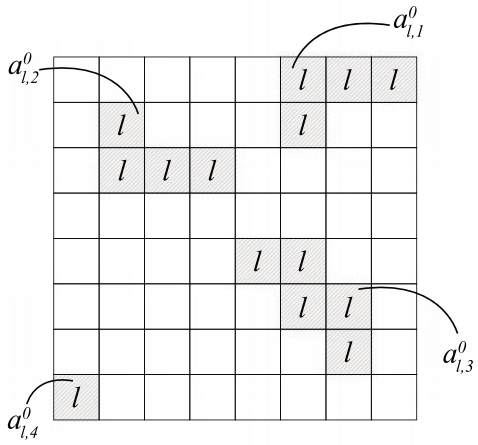


Figure 3.1: Partition A_l^0 consisting of 4 disjoint subsets at iteration 0.

Next, we consider the expanded set partition A_l^{i+1}, corresponding to the application of a set operator f at iteration i:

$$\begin{aligned} A_l^{i+1} &= f(A_l^i) \\ \vdots &\quad\ \vdots \\ A_l^1 &= f(A_l^0) \end{aligned} \tag{3.5}$$

where $f(A)$ is a function that expands set A.

Therefore, given I_l iterations of the expanded class set partitions, we have L equiva-

lence classes as follows:

$$\begin{array}{llll} \{A_1^0, & A_1^1, & ..., & A_1^{I_1}\} \quad \textit{Equivalence class } 1 \\ \{A_2^0, & A_2^1, & ..., & A_2^{I_2}\} \quad \textit{Equivalence class } 2 \\ \vdots & \vdots & & \vdots \\ \{A_L^0, & A_L^1, & ..., & A_L^{I_L}\} \quad \textit{Equivalence class } L \end{array} \tag{3.6}$$

A candidate set morphology-preserving operation f based on mathematical morphology is the dilation [98]:

$$f(A_l^i) = dil^B(A_l^i) \tag{3.7}$$

by a suitable structuring element B. A potential result for iteration 1 is shown in Figure 3.2, assuming that iteration 0 is shown in Figure 3.1, where in this specific case $B = \{(0,0),(-1,0),(1,0),(0,-1),(0,1)\}$.

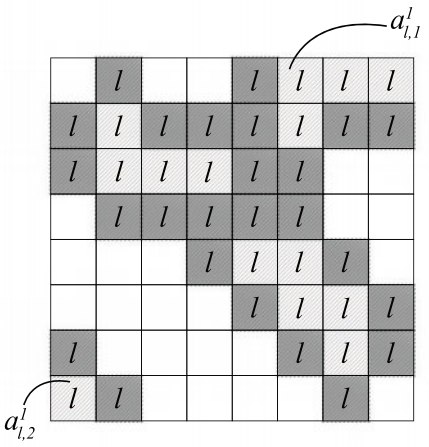


Figure 3.2: Set partition A_l^1 consisting of 2 disjoint subsets at iteration 1, where the dark grey shaded pixels represent the elements that are included in the set at iteration 1.

In a dilator classifier, all A_l^0 can be expanded simultaneously, where intersections of labels are erased when they occur. The result of the dilation of l classes until they cover the classification space $\mathbb{S}$ is a classification model.

The operation f may be any other operation apart from strict dilations. For instance, it could be a combination of morphological operations such as dilations and erosions, a pseudo-random or random function operating on the set partition, etc. Opening and closings can be used to modify initial data points (e.g., noise removal) such as with the BMCA algorithms.

In the case of p-dimensional set partitions A_l^0, patterns are represented in a p-dimensional hypercube or grid. Each original set partition A_l^0 is represented by a unique class label l, and an appropriate set operator (such as a dilation shown in Equation 3.7), is applied to produce the expanded set partition A_l^i at iteration i. The set expansion process is controlled by a convergence function, which monitors a performance measure, e.g., classification error. The evolution of a class set partition at iteration i is terminated according to the convergence function, and the membership of the class set partition is reset to that at iteration $i-1$. In what follows, we provide the definition of a Morphological Classifier (MC):

Definition Morphological Classifier (MC): Assume a partition $\mathbb{S}$ of $\mathbb{R}^p$ and L set partitions $A_l^{I_l}$ such that:

$$\bigcup_{l=1}^{L} A_l^{I_l} = \mathbb{S} \tag{3.8}$$

where I_l corresponds to the final iteration for class l, and each of the set partitions is the result of morphology-preserving set expansion operations.

A morphological classifier M is a p-dimensional hypercube or grid, where the value of each element refers to the predicted label of an unclassified pattern y.

$$\hat{c}(\mathrm{y}) = l : \mathrm{y} \in A_l^{I_l}, l \in \{1, ..., L\} \tag{3.9}$$

The evolution of set partitions $A_l^{I_l}$ is governed by a convergence function $\phi()$.

Given a validation dataset with m instances, $Y = \{\mathrm{y}_1, ..., \mathrm{y}_m\} \subset \mathbb{S}$, which is a multiset with corresponding labels $c(\mathrm{y}_1), ..., c(\mathrm{y}_m) \in \{1, ..., L\}$, its accuracy rate is given by:

$$Accuracy = \frac{|\{\mathrm{y} \in Y : \hat{c}(\mathrm{y}) = c(\mathrm{y})\}|}{|Y|} \tag{3.10}$$

where $\hat{c}(\mathrm{y})$ represents the predicted class of instance y (as shown in Equation 3.3) and $|Y|$ stands for the cardinality of set Y.

Theorem 3.2.1 proposes a possible methodology for the expansion of the set partitions. The fundamental concept of the theorem is that it is possible to exploit redundancy, i.e., the complement of the union of $L-1$ set partitions represents one of the set partitions, which reduces memory and computational requirements.

However, with L set partitions, there are L possible options for the complement set partition, i.e., which candidate complement set partition leads to the best accuracy. Theorem 3.2.1 proves that the selection of the complement partition affects accuracy.

Theorem 3.2.1 *Impact of complement set partition A_q on classification performance:*

Consider the L expanded set partitions A_l^i at iteration i, $l \in \{1, ..., L\}$,

$$A_q^i = \mathbb{S} \setminus \bigcup_{l=1}^{L} \begin{cases} A_l^i, & if\ l \neq q \\ \emptyset, & otherwise \end{cases} \tag{3.11}$$

subject to

$$\mathbb{S} = \bigcup_{l=1}^{L} A_l^i, \mathbb{S} \subset \mathbb{R}^p \tag{3.12}$$

then, there is a set partition $A_q^{I_l}$, $q \in \{1, ..., L\}$, at final iteration I_l, that maximizes accuracy for a specific classification problem.

Proof *Equation 3.11 introduces the use of the complement of a set partition in the expansion process, which reduces memory and processing time requirements. For instance, in binary classifications, the partitions are: A_1^i and A_2^i. The expansion is performed in one of the two partitions, while the other corresponds to the complement of the first partition.*

Assuming that X_l is a multiset of instances with labels equal to l, then the accuracy rate for the binary case is given by:

$$Accuracy = \frac{|X_1 \bigcap A_1^{I_1}| + |X_2 \bigcap A_2^{I_2}|}{|X|} \tag{3.13}$$

Assuming $A_q^{I_q} = A_2^{I_2}$, then $|X_2 \bigcap A_2^{I_2}|$ is equivalent to $|X_2 \setminus A_1^{I_1}|$, as $A_2^{I_2}$ is the complement set partition ($A_2^{I_2} = \mathbb{S} \setminus A_1^{I_1}$). Thus, the accuracy for $A_q^{I_q} = A_2^{I_2}$ is given by:

$$Accuracy_{A_q^{I_q} = A_2^{I_2}} = \frac{|X_1 \bigcap A_1^{I_1}| + |X_2 \setminus A_1^{I_1}|}{|X|} \tag{3.14}$$

The previous accuracy concerns to $A_q^{I_q} = A_2^{I_2}$. In the other case, where $A_q^{I_q} = A_1^{I_1}$, instead of expanding $A_1^{I_1}$, $A_2^{I_2}$ is expanded, which provides the following accuracy rate:

$$Accuracy_{A_q^{I_q}=A_1^{I_1}} = \frac{|X_2 \bigcap A_2^{I_2}| + |X_1 \setminus A_2^{I_2}|}{|X|} \tag{3.15}$$

We seek to prove that the choice of A_q^i has an impact on the accuracy. For that to be true, we assume that Equation 3.16 is true and derive contradiction. If this does not hold for at least a single case, then our theorem is proved.

$$\begin{array}{rcl} Accuracy_{A_q^{I_q}=A_2^{I_2}} & = & Accuracy_{A_q^{I_q}=A_1^{I_1}} \\ \frac{|X_1 \bigcap A_1^{I_1}| + |X_2 \setminus A_1^{I_1}|}{|X|} & = & \frac{|X_2 \bigcap A_2^{I_2}| + |X_1 \setminus A_2^{I_2}|}{|X|} \end{array} \tag{3.16}$$

Let us assume a similar case for both set accuracy computations, where $A_1^{I_1}$ and $A_2^{I_2}$, respectively, are expanded to the point where they cover the space $\mathbb{S}$ completely, such that $A_1^{I_1} = A_2^{I_2} = \mathbb{S}$. This implies $X_2 \setminus A_1^{I_1} = \emptyset$, $X_1 \setminus A_2^{I_2} = \emptyset$, $X_1 \bigcap A_1^{I_1} = X_1$ as well as $X_2 \bigcap A_2^{I_2} = X_2$. Therefore:

$$\begin{array}{rcl} \frac{|X_1 \bigcap A_1^{I_1}| + |X_2 \setminus A_1^{I_1}|}{|X|} & = & \frac{|X_2 \bigcap A_2^{I_2}| + |X_1 \setminus A_2^{I_2}|}{|X|} \\ |X_1 \bigcap A_1^{I_1}| + |X_2 \setminus A_1^{I_1}| & = & |X_2 \bigcap A_2^{I_2}| + |X_1 \setminus A_2^{I_2}| \\ |X_1 \bigcap A_1^{I_1}| + |\emptyset| & = & |X_2 \bigcap A_2^{I_2}| + |\emptyset| \\ |X_1 \bigcap A_1^{I_1}| & = & |X_2 \bigcap A_2^{I_2}| \\ |X_1| & = & |X_2| \end{array} \tag{3.17}$$

$|X_1| = |X_2|$ is a contradiction as X_1 and X_2 can be different multisets. Therefore, the accuracy can be affected by the choice of the complement set partition. ■

As Theorem 3.2.1 indicates, accuracy rates may vary depending on the chosen complement partition. We recommend the use of brute force approaches in this case. Binary classification problems were tested in the course of this research, and it is possible to assume that accuracy varies slightly over partitions, in general. We suspect that this can be assumed in general case scenarios.

In addition, we assume that the classification space is a discrete space. Depending on the dataset being used, it can represent a finer grain of information, i.e., its grid elements could be smaller, or it each one could encapsulate several instances, i.e., grid elements would be larger. Datasets can be discretized/quantized in a way that no information is actually lost.

3.2.1 Morphological k-NN (MkNN)

In this section, we present an algorithm that is similar to k-NN while still conforming to the theoretical definitions of Section 3.2. This algorithm generates a classification model, which does not require re-training and therefore is not a lazy classifier. However, as in the traditional k-NN, an unlabelled instance y has its surroundings examined until the algorithm checks k other instances. Finally, it classifies y as the majority class among visited neighbouring instances.

Formally, given an unlabelled instance y, its neighbouring instances are visited so that at iteration i, labelled instances that are at a distance $\leq i$ from y are visited. If at any iteration i, the number of visited labelled instances is $\geq k$, where k is an input parameter as in the k-NN, then the unlabelled instance y is said to be of the same class as the majority of the visited labelled neighbouring instances. The f operation in this case is given by:

$$A_l^{i+1} = f(A_l^i) = A_l^i \cup \{\mathrm{y} \in \mathbb{S} : mode(\{\mathrm{x} \in X : d(\mathrm{x},\mathrm{y}) \leq i\}) = l\} \tag{3.18}$$

where d is a distance function and $mode()$ returns the most frequent label in the multiset. The algorithm proceeds by analysing and examining the nearest labelled instances. For a single unlabelled position y in grid or space $\mathbb{S}$, the surrounding labelled instances x are examined. This whole process is repeated for every possible unlabelled position in the grid before the algorithm reaches convergence, i.e., it terminates when k nearest instances have been visited. The convergence function for class l is given by:

$$\phi(A_l^i) = \begin{cases} 1, & if \ \sum_{\mathrm{y}}^{\mathrm{y}\in\mathbb{S}} \phi_a(\mathrm{y}) = |\mathbb{S}| \\ 0, & otherwise \end{cases} \tag{3.19}$$

which is subject to

$$\phi_a(\mathrm{y}) = |\{\mathrm{x} \in X : d(\mathrm{x},\mathrm{y}) \leq i\}| \geq k \tag{3.20}$$

where $|\mathbb{S}|$ indicates the total size of the subset or grid $\mathbb{S}$, i.e., the total amount of elements that represent the grid. Finally, the labelling function of MkNN algorithm conforms to Equation 3.9. The pseudo-code for constructing its classification model is shown in Algorithm 1.

Algorithm 1: Construction of the MkNN classification model.

Data: k and γ are input parameters, where k stands for how many neighbouring instances should be visited, γ stands for a weight applied to the central grid position and σ represents the maximal number of iterations. $T_1, ..., T_L$ are counters for the classes, X is the training dataset, A_l^0 respects Equation 3.4 and $c(\mathrm{x})$ returns the class label of instance x.

1 **begin**
2 **for** *each possible* y *position in grid* $\mathbb{S}$ **do**
3 $k_{aux} \leftarrow 0$; $i \leftarrow 0$;
4 **for** *each class* l, *such that* $1 \leq l \leq L$ **do** $T_l \leftarrow 0$;
5 **if** $\mathrm{y} \in X$ **then** $T_{c(\mathrm{y})} \leftarrow \gamma$;
6 **while** $k_{aux} < k$ **do**
7 **for** *each instance* $\mathrm{x} \in X : d(\mathrm{y}, \mathrm{x}) = i$ **do**
8 $T_{c(\mathrm{x})} \leftarrow T_{c(\mathrm{x})} + 1$;
9 $k_{aux} \leftarrow k_{aux} + 1$;
10 $i \leftarrow i + 1$;
11 **if** $\sigma < i$ **then** break;
12 $maxl \leftarrow$ receives the l that maximizes T_l;
13 $A_{maxl}^{i+1} \leftarrow (\bigcup_{i2=0}^{i2<i+1} A_{maxl}^{i2}) \cup \{\mathrm{y}\}$;
14 Return $A_1^{I_l}, ..., A_L^{I_l}$;

3.2.2 Morphological Dilator Classifier (MDC)

The Morphological Dilator Classifier inherits more properties of mathematical morphology than MkNN. This algorithm dilates positions in grid $\mathbb{S}$ until a termination criterion is reached. The f operation for the MDC is governed by a rule β that indicates the orientation in which the instances are dilated. Function f is then given by:

$$A_l^{i+1} = f(A_l^i) = dil(A_l^i, B(d, i, \beta)) \tag{3.21}$$

where the operator dil refers to a p-dimensional dilation of the expanded set partition A_l^i at iteration i by the p-dimensional structuring element B. The structuring element B is a set whose structure is defined by the choice of distance d, the distance measure i (or iteration counter) and the orientation factor β.

Examples of expansion rules for β, which alter the orientation of the structuring element and hence the expansion of the associated set partition for the case of two dimensions, are:

$$\beta = \begin{cases} e_1 \geq -|e_2| & : to\ the\ left \\ e_2 \geq |e_1| & : to\ the\ bottom \\ e_1 \leq |e_2| & : to\ the\ right \\ e_2 \leq -|e_1| & : to\ the\ top \end{cases} \tag{3.22}$$

where e_1 represents the x coordinate of e and e_2 its y coordinate. Combinations of these rules are also possible. Different expansions and p-dimensional rules can also be applied. For the MDC implementation of this work, we consider these 4 different orientations and any combination of them, resulting in a total of 16 different growth patterns.

For more detail, the f function of MDC is given by:

$$A_l^{i+1} = f(A_l^i) = A_l^i \cup \{\mathrm{y} \in \mathbb{S} : d(\mathrm{x}, \mathrm{y}) = i, \mathrm{x} \in A_l^i, \beta\} \tag{3.23}$$

Besides, the convergence function of MDC is given by:

$$\phi(A_l^i) = \begin{cases} 1, & if\ P_{err}(A_l^i) < t_{err} \vee i \geq \sigma \\ 0, & otherwise \end{cases} \tag{3.24}$$

where P_{err} is a classification error measure and t_{err} is a error threshold, indicating the

maximal accepted error prior to the classifier terminating training. σ forces expansions to terminate if i exceeds σ. $\vee$ represents the Boolean *or* operation. The individual error $P_{instErr}(\mathrm{y})$ of an instance y is given by:

$$P_{instErr}(\mathrm{y}) = \begin{cases} 1, & if\ \tau T_l < T_u, u \in \{1, ..., L\}, u \neq l \\ 0, & otherwise \end{cases} \tag{3.25}$$

where T_l is a counter of neighbouring instances of y that have been engulfed by the expansion and belong to class l. τ is a scaling factor that, such as in GRASP [72], allows for the expansion to continue even if $T_l < T_u$. The total error of the set partition is the sum of $P_{instErr}$ such that:

$$P_{err}(A_l^i) = \sum_{\mathrm{y}}^{\mathrm{y} \in A_l^i} P_{instErr}(\mathrm{y}) \tag{3.26}$$

The pseudo-code for generating the MDC classification model is shown in Algorithm 2.

Algorithm 2: Construction of the MDC classification model.

Data: γ, τ, β and σ are input parameters. γ stands for a weight applied to the central instance, τ controls the convergence as shown in Equation 3.25, β a orientation or expansion altering rule, as shown in Equation 3.22, and σ is a threshold for the number of partition set expansions. $T_1, ..., T_L$ are counters for the classes $1...L$, X is the training dataset and A_l^0 respects Equation 3.4. q indicates the complement set partition.

```
1  begin
2      for each class l, such that 1 ≤ l ≤ L, l ≠ q do
3          i ← 0; for each instance e ∈ A_l^i do
4              for each class l_two, such that 1 ≤ l_two ≤ L do T_{l_two} ← 0;
5              T_{c(e)} ← γ;
6              while P_instErr(e) = 0 and i < σ do
7                  for each instance x ∈ X : d(e, x) ≤ i do
8                      T_{c(x)} ← T_{c(x)} + 1;
9                  if P_instErr(e) = 0 then
10                     A_l^{i+1} = A_l^i ∪ {y ∈ 𝕊 : d(e, y) ≤ i, y ∉ A_{l2}^i : l2 ∈ {1, ..., L}};
11                 i ← i + 1;
12     Apply the complement operation to set partition A_q^{I_q} as described in Equation
       3.11;
13     Return A_1^{I_l}, ..., A_L^{I_l};
```

3.2.2.1 Graphical Effects of Structuring Elements

Algorithms for morphological classification were addressed. However, differences on classification models that originate from the choice of different structuring elements have not yet been shown. Different structuring elements or rules β can modify the expansion progress of the partition sets dramatically. Figure 3.3 shows an input dataset containing a single label that we wish to dilate.

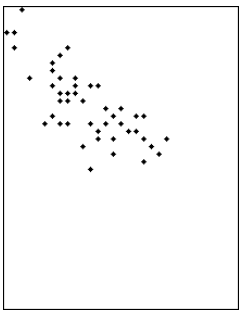
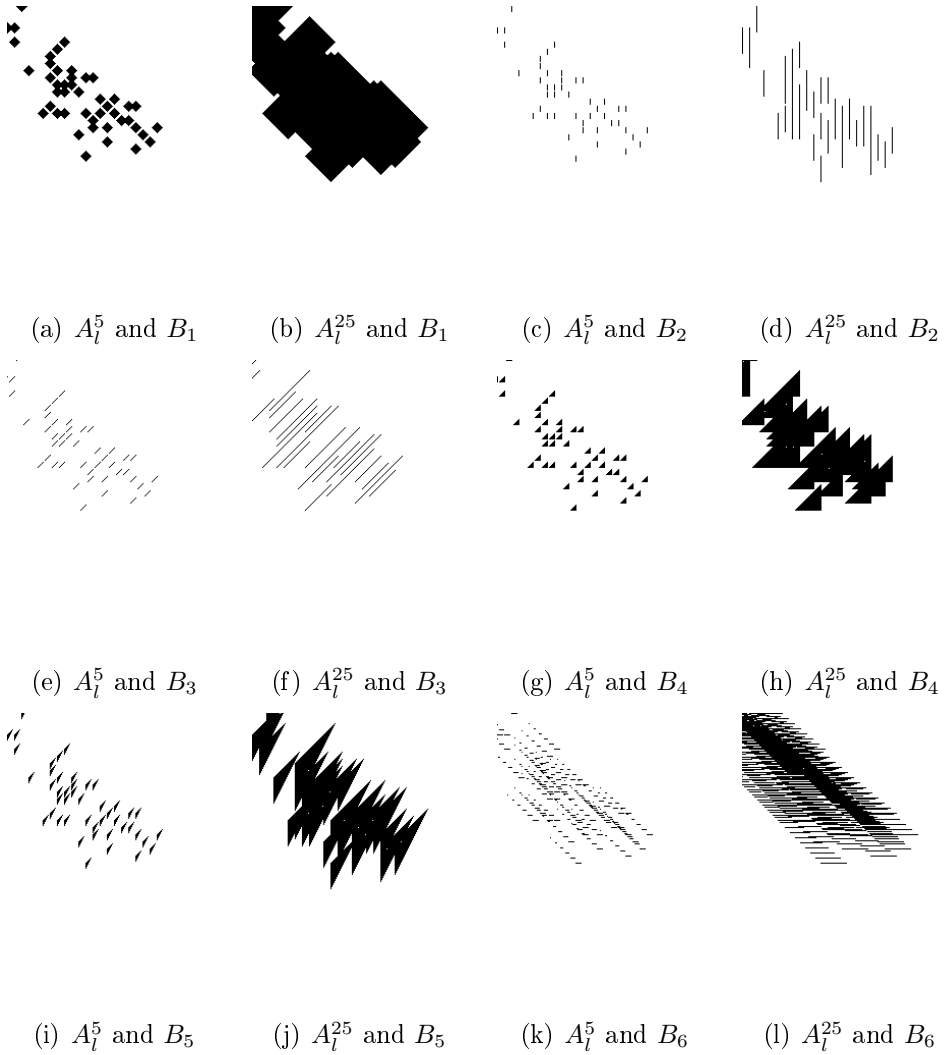


Figure 3.3: Instances of an input two dimensional dataset with class label equal to l.

At different iterations, e.g., 5 and 25, we would have the following sets A_l^5 and A_l^{25} for the given class l. In these examples, the used structuring elements were: $B_1 = \{(0,0),(-1,0),(1,0),(0,-1),(0,1)\}$, $B_2 = \{(0,0),(-1,0)\}$, $B_3 = \{(0,0),(1,-1)\}$, $B_4 = \{(0,0),(1,0),(0,1)\}$, $B_5 = \{(0,0),(1,0),(1,-1)\}$, $B_6 = \{(0,0),(-1,0),(5,5)\}$, respectively.



(a) A_l^5 and B_1 (b) A_l^{25} and B_1 (c) A_l^5 and B_2 (d) A_l^{25} and B_2

(e) A_l^5 and B_3 (f) A_l^{25} and B_3 (g) A_l^5 and B_4 (h) A_l^{25} and B_4

(i) A_l^5 and B_5 (j) A_l^{25} and B_5 (k) A_l^5 and B_6 (l) A_l^{25} and B_6

Figure 3.4: Examples of the f function being a dilation with varying structuring elements at iterations 5 and 25.

Different structuring elements lead to very distinct classification models, as illustrated in Figure 3.4. This fact can be explored respecting particularities of each dataset. In 3.4-(l), for instance, it is clear that the classification model does not need to be "connected". Brute force is still recommended for the general case. For a even broader case, a symmetric structuring element such as the cross-shaped one or ring can be used.

3.2.3 Rodrigues Distance

The two classifier implementations that were presented up to this point consider p-dimensional multi-labelled data. In the experiments (Chapter 4), however, we decided to process p-dimensional datasets as combinations of 2-dimensional models. Several 2D "weak" learners can be created from p-dimensional data and combined to form a

p-dimensional classification model.

2D models are faster to compute and more efficient to store. Not every pairwise combination of attributes has to be explored. Even if necessary, it is still more efficient (proved in Theorem 3.2.3). Heuristics can be used to eliminate some of these combinations. In this subsection, we introduce a distance that is very efficient in GPUs or in low level environments for 2D neighbourhood iterations. The order in which the neighbours of a pixel y are visited in both MkNN and MDC algorithms is shown in Figure 3.5, for the upper-right quadrant. The 3 remaining quadrants (left, bottom and bottom-left) respect the same ordering.

15	16	17	18	19	20
10	11	12	13	14	19
6	7	8	9	13	18
3	4	5	8	12	17
1	2	4	7	11	16
y	1	3	6	10	15

Figure 3.5: Upper-right quadrant of the proposed distance metric.

This iteration, however, can be performed using different approaches. If we consider the Chebyshev distance, the solution is straightforward in terms of algorithmic logic. A loop iterates through the two lines above and below the central pixel and through the columns at its left and right. At each increment of the distance, the lines and columns are shifted by 1. Algorithm 3 illustrates the concept. After convergence, N_d contains the pixels at distances d from the central pixel (x_c, y_c).

It is slightly more difficult to perform neighbourhood iterations using the Manhattan distance. Manhattan can be seen as the Chebyshev distance rotated by 45 degrees in $\mathbb{R}^2$. However, the implementation requires a few more operations and variables, which slightly increases the overall processing time. Algorithm 4 describes the process.

Using any other distance than Chebyshev and Manhattan requires much more effort. The Euclidean distance requires a quantization/approximation process. The distances can be pre-computed and stored in a kernel or processed in real time. However, storing large amounts of data in kernels consume a significant amount of memory when computing large distances. Accessing this data also impacts negatively on the overall processing performance. In the GPU, for instance, these kernels would have to be located in global

Algorithm 3: Neighbourhood iteration using the Chebyshev distance.

Data: I stands for an arbitrary image, $I(x, y)$ represents its pixel value at position (x, y). D represents the maximal distance to be computed and N_d is a set that contains all the pixels at distance d from central pixel (x_c, y_c).

```
1 d ← 1;
2 while d < D do
3     for (l ← −d; l ≤ d; l ← l + 1) do
4         N_d ← N_d ∪ I(x_c + l, y_c − d);
5         N_d ← N_d ∪ I(x_c + l, y_c + d);
6         if l ≠ d and l ≠ −d then // Avoiding duplicates at corners
7             N_d ← N_d ∪ I(x_c − d, y_c + l);
8             N_d ← N_d ∪ I(x_c + d, y_c + l);
9     d ← d + 1;
```

Algorithm 4: Neighbourhood iteration using the Manhattan distance.

Data: I stands for an arbitrary image, $I(x, y)$ represents its pixel value at position (x, y). D represents the maximal distance to be computed and N_d is a set that contains all the pixels at distance d from central pixel (x_c, y_c).

```
1  d ← 1; x_a ← 0; y_a ← 0;
2  while d < D do
3      for (g ← 0; g < 4; g ← g + 1) do
4          switch g do
5              case 0 do x_a ← x_c; y_a ← y_c + d;
6              case 1 do x_a ← x_c + d; y_a ← y_c;
7              case 2 do x_a ← x_c; y_a ← y_c − d;
8              case 3 do x_a ← x_c − d; y_a ← y_c;
9          for (l ← 0; l ≤ d; l ← l + 1) do
10             N_d ← N_d ∪ I(x_a, y_a);
11             switch g do
12                 case 0 do  x_a ← x_a + 1; y_a ← y_a − 1;
13                 case 1 do  x_a ← x_a − 1; y_a ← y_a − 1;
14                 case 2 do  x_a ← x_a − 1; y_a ← y_a + 1;
15                 case 3 do  x_a ← x_a + 1; y_a ← y_a + 1;
16     d ← d + 1;
```

memory, whose access is slow. Computing kernels in real time is also very inefficient.

Algorithm 5 illustrates an arguably efficient way of iterating through the neighbourhood of the central pixel (x_c, y_c) using the Euclidean distance without storing the respective distances in a pre-built kernel. Two early breaks are employed to speed up the computation.

Algorithm 5: Neighbourhood iteration using the Euclidean distance.

Data: I stands for an arbitrary image, $I(x, y)$ represents its pixel value at position (x, y). D represents the maximal distance to be computed and N_d is a set that contains all the pixels at distance d from central pixel (x_c, y_c), and *abs* represents the absolute integer value.

```
1  d ← 1; x_a ← 0; y_a ← 0;
2  while d < D do
3      for (ls ← 0; ls < d/2; ls ← ls + 1) do
4          foundFirst ← false;
5          for (l ← d; l ≥ 0; l ← l − 1) do
6              finished ← false;
7              for (ln ← l; ln ≥ −l; ln ← ln − (2 ∗ l)) do
8                  x_a ← x_c + ln; y_a ← y_c − d + ls;
9                  if δ then
10                     N_d ← N_d ∪ I(x_a, y_a);
11                     foundFirst ← true;
12                 else finished ← foundFirst;
13                 y_a ← y_c + d − ls;
14                 if abs(√((x_c − x_a)² + (y_c − y_a)²)) = d then N_d ← N_d ∪ I(x_a, y_a);
15                 if ln ≠ d then // duplicates
16                     x_a ← x_c − d + ls; y_a ← y_c + ln;
17                     if abs(√((x_c − x_a)² + (y_c − y_a)²)) = d then N_d ← N_d ∪ I(x_a, y_a);
18                     x_a ← x_c + d − ls;
19                     if abs(√((x_c − x_a)² + (y_c − y_a)²)) = d then N_d ← N_d ∪ I(x_a, y_a);
20                 if ln = 0 then break;
21             if finished then break;
22     d ← d + 1;
```

On average, the Chebyshev distance (Algorithm 3) iterates neighbourhoods up to distance $D = 2500$ in 0.098 seconds. D represents the size of the discrete neighbourhood around the pixel that is being iterated. The Manhattan distance (Algorithm 4), on the other hand, takes approximately 0.146 seconds to iterate through the same neighbourhood. The Euclidean distance (Algorithm 5), at last, requires a total of 36.125 seconds, which is significantly worse.

When it comes to optimization, especially in GPU computing, these differences on the

performance impact on the overall processing times even more. Besides, the Manhattan distance does not add any information to the neighbourhood iteration when compared to the Chebyshev distance. As previously stated, the Manhattan distance is identical to the Chebyshev distance, but rotated by 45 degrees in $\mathbb{R}^2$.

On the other hand, using the Euclidean distance implies a huge computational burden, where its implementation is approximately 368 times slower than Chebyshev for $D = 2500$, and it gets even worse as D increases. This work proposes an intermediate distance metric between Chebyshev and Euclidean that adds information, as opposed to Manhattan, and is far more efficient when it comes to neighbourhood iterations.

This distance (its mathematical formulation is shown in Equation 3.27) was established while attempting to improve the neighbourhood iterations of morphological classifiers in GPUs, which heavily rely on low level operations. As previously stated, a metric closer to the Euclidean distance was desired, but the reported time burden was incompatible with practical cases. Algorithm 6 shows the iteration process of the proposed distance.

Algorithm 6: Neighbourhood iteration using the proposed/Rodrigues distance.

Data: I stands for an arbitrary image, $I(x, y)$ represents its pixel value at position (x, y). D represents the maximal distance to be computed and N_d is a set that contains all the pixels at distance d from central pixel (x_c, y_c).

```
1  d ← 1; d_a ← d;
2  while d < D do
3      N_da ← N_da ∪ I(x_c + d, y_c);
4      N_da ← N_da ∪ I(x_c − d, y_c);
5      N_da ← N_da ∪ I(x_c, y_c + d);
6      N_da ← N_da ∪ I(x_c, y_c − d);
7      for (l ← 1; l ≤ d; l ← l + 1) do
8          N_(da+l) ← N_(da+l) ∪ I(x_c + d, y_c − l);
9          N_(da+l) ← N_(da+l) ∪ I(x_c + d, y_c + l);
10         N_(da+l) ← N_(da+l) ∪ I(x_c − d, y_c − l);
11         N_(da+l) ← N_(da+l) ∪ I(x_c − d, y_c + l);
12         if l ≠ d then
13             N_(da+l) ← N_(da+l) ∪ I(x_c − l, y_c + d);
14             N_(da+l) ← N_(da+l) ∪ I(x_c + l, y_c + d);
15             N_(da+l) ← N_(da+l) ∪ I(x_c − l, y_c − d);
16             N_(da+l) ← N_(da+l) ∪ I(x_c + l, y_c − d);
17     d_a ← d_a + d + 1;
18     d ← d + 1;
```

This proposed iteration (Algorithm 6) achieves almost Chebyshev-like time perfor-

mances and obtains an iteration fashion that is closer to the Euclidean. The result of the proposed distance in $\mathbb{R}^2$ for $p = 1$ is an octagon (Chebyshev and Manhattan distances produce squares). That is, the octagon is much closer to the circular pattern produced by the Euclidean distance than the squares produced by the Manhattan and Chebyshev distances. Table 3.1 shows the processing times obtained with each one of the algorithms (Algorithms 3-6) as the size of the neighbourhood D is increased. These results were obtained using Java 7 and an Intel i7-7700HQ, clocked at 2.8 GHz averaged over 100 runs.

Table 3.1: Processing times (s) for each distance (Algorithms 3-6).

Sizes (D)	Chebyshev	Manhattan	Euclidean	**Rodrigues**
500	0.0022	0.0024	0.3069	0.0022
1000	0.0109	0.0116	2.3290	0.0115
1500	0.0295	0.0407	7.8285	0.0318
2000	0.0583	0.0829	18.5112	0.0624
2500	0.0985	0.1460	36.1253	0.1050
3000	0.1517	0.2228	62.2829	0.1599
3500	0.2377	0.3397	98.8083	0.2454
4000	0.3031	0.4672	147.6250	0.3166
4500	0.4030	0.5923	204.2881	0.4243

Table 3.1 demonstrates that the proposed distance is in fact faster than the Manhattan and Euclidean distances. On average, it was 1.3 times faster than Manhattan and 329.5 times faster than Euclidean. In what follows, a formal mathematical definition of this distance is presented.

3.2.3.1 Definition

The proposed distance is a combination of Chebyshev and Minkowski distances, weighted by w_1 and w_2, as shown in Equation 3.27. As w_1 increases in regard to w_2, the distance becomes more like Minkowski. On the contrary, when w_2 increases it converges towards Chebyshev. Algorithm 6 considers $w_1 = w_2 = p = 1$.

Definition The proposed distance is defined as:

$$\begin{aligned} d_{w_1,w_2,p}(x,y) &= w_1\, d_p(x,y) + w_2\, d_\infty(x,y) \\ &\vdots \\ d_{w_1,w_2,p}(x,y) &= w_1 \sqrt[p]{\textstyle\sum_{i=1}^{n} |x_i - y_i|^p} + w_2 \max_{i=1}^{n} |x_i - y_i| \end{aligned} \tag{3.27}$$

As two metrics are being summed up, it is straightforward to infer from the formulation that as long as $w_1, w_2 > 0$ and $p \geq 1$, all the metric conditions are satisfied, including the triangle inequality.

In what follows, Theorem 3.2.2 demonstrates in detail how the metrics conditions are satisfied for $d_{w_1,w_2,p}(x,y)$.

Theorem 3.2.2 *$d_{w_1,w_2,p}(x,y)$ is a metric given that $p \geq 1, w_1 > 0$ and $w_2 > 0$.*

Proof *① $d_{w_1,w_2,p}(x,y) \geq 0 \rightarrow d_p(x,y)$ and $d_\infty(x,y)$ are metrics themselves and therefore provide values equal or greater than 0, their sum is also equal or greater than 0.*

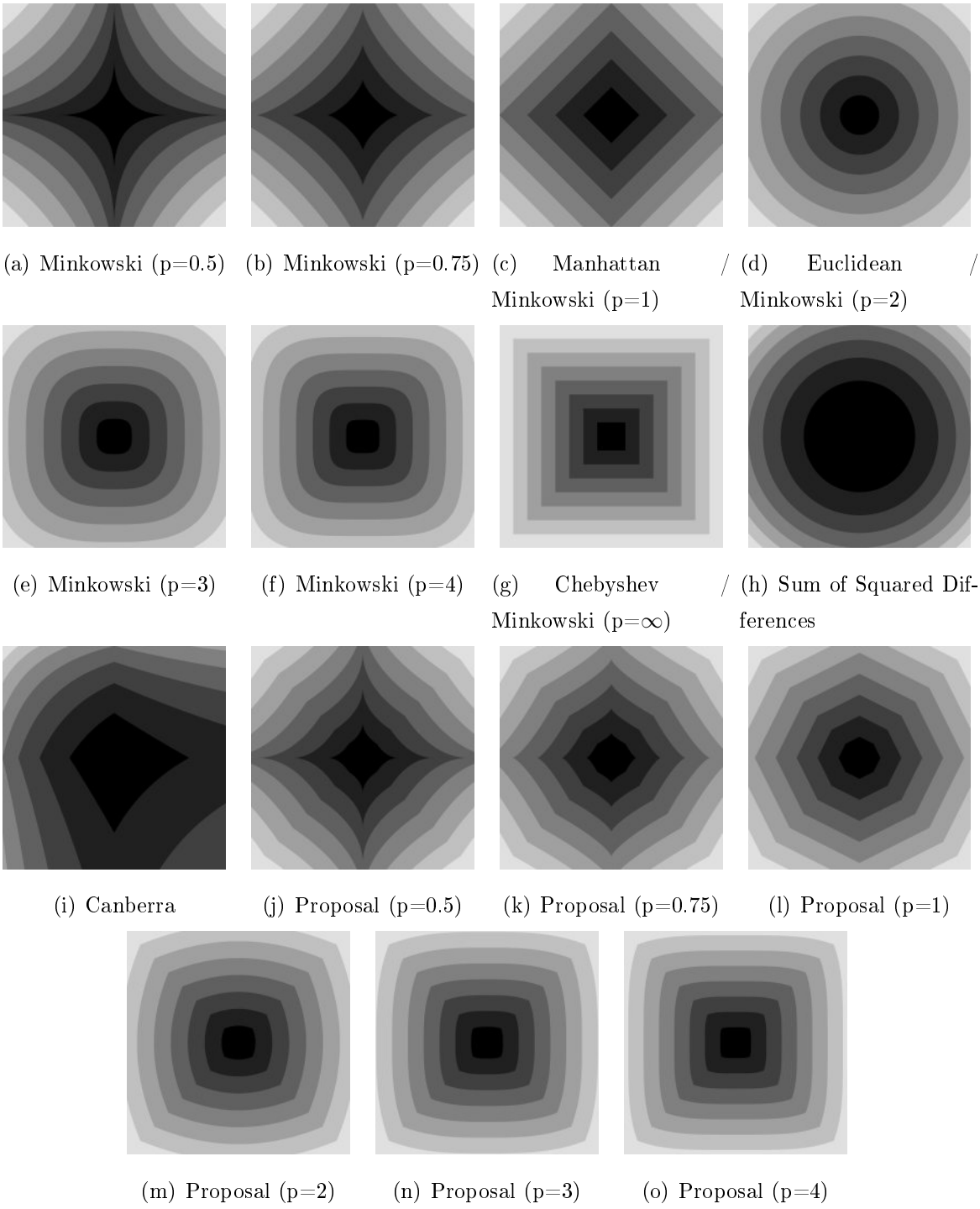
② $d_{w_1,w_2,p}(x,y) = 0 \Leftrightarrow x = y \rightarrow d_p(x,y)$ and $d_\infty(x,y)$ both return 0 if and only if $x = y$. The sum of these two results in 0 only if both are 0.

③ $d_{w_1,w_2,p}(x,y) = d_{w_1,w_2,p}(y,x) \rightarrow 0$, It is known a priori that both metrics $d_p(x,y)$ and $d_\infty(x,y)$ satisfy ③. Therefore, $d_p(x,y) = d_p(y,x)$ and $d_\infty(x,y) = d_\infty(y,x)$, which proves that $w_1\, d_p(x,y) + w_2\, d_\infty(x,y) = w_1\, d_p(y,x) + w_2\, d_\infty(y,x)$.

④ $d_{w_1,w_2,p}(x,z) \leq d_{w_1,w_2,p}(x,y) + d_{w_1,w_2,p}(y,z) \rightarrow$ Once again, as both $d_p(x,z) \leq d_p(x,y) + d_p(y,z)$ and $d_\infty(x,z) \leq d_\infty(x,y) + d_\infty(y,z)$ are true, consequently, $w_1\, d_p(x,z) + w_2\, d_\infty(x,z) \leq w_1\, d_p(x,y) + w_2\, d_\infty(x,y) + w_1\, d_p(y,z) + w_2\, d_\infty(y,z)$ is also satisfied. ■

3.2.3.2 Graphical Analysis in $\mathbb{R}^2$

Figure 3.6 shows how the proposed distance compares to other distances in $\mathbb{R}^2$. The proposed distance is in fact a mixture of Chebyshev and Manhattan (for p = 1), as shown in Figure 3.6-(l). It is possible to argue that the distance is an intermediate step between Manhattan/Chebyshev and Euclidean, and therefore it adds information in relation to Manhattan and Chebyshev. However, it requires far less processing power (as shown in Algorithm 6) in regard to Euclidean or even Manhattan. In the cases where $p < 1$, the respective distances are not metrics. The condition $w_1 = w_2 = 1$ is respected for all the distances in this figure.



(a) Minkowski (p=0.5) (b) Minkowski (p=0.75) (c) Manhattan / Minkowski (p=1) (d) Euclidean / Minkowski (p=2)

(e) Minkowski (p=3) (f) Minkowski (p=4) (g) Chebyshev / Minkowski (p=∞) (h) Sum of Squared Differences

(i) Canberra (j) Proposal (p=0.5) (k) Proposal (p=0.75) (l) Proposal (p=1)

(m) Proposal (p=2) (n) Proposal (p=3) (o) Proposal (p=4)

Figure 3.6: Distances in $\mathbb{R}^2$. The distances are computed from the central element in the image. Lighter shades of gray indicate greater distances.

3.2.3.3 Performance Analysis Using k-NN

In this subsection, we present a practical experiment using 33 numerical datasets from the UCI repository. This encompass all the entirely numerical datasets in their repository excluding kdd-JapaneseVowels and Liver Disorders. We varied the distance measurements

and the k variable of the traditional k-NN classifier to generate the results shown in Table 3.2.

Categorical attributes are widely used with k-NN and other classifiers that are supposedly made to work with numerical/continuous values. The commonly employed assumption is that if the categorical value is equal, the distance between these two instances is said to be 1. Otherwise, the categorical distance is set to 0. Although this assumption works well and enables the processing of categorical datasets by numerical classifiers, it could bias the experiments. If an arbitrary dataset contains 9 categorical and 1 numerical attributes, the distance measurement would be applied in just one of the attributes (1 dimensional). These assumptions could more frequently improve the performance of a certain distance in detriment of others by chance. In order to avoid biasing the results, categorical datasets were disregarded from this experiment.

The first three columns of Table 3.2 show the mean accuracy, true positive and true negative rates achieved by each distance. Accuracy is defined as the number of true positives and true negatives divided by the total population. The last five columns concern to the Mean k, Max k, P-value, how many times the accuracy obtained with the distance was better than the average accuracy, and the amount of occasions where the accuracy obtained with the respective distance was the best one (equal to the maximal obtained accuracy).

The k variable was tested with values from 1 to 200 for each combination of distance and dataset. The value for k that achieved the best accuracy was selected. The fifth column (Mean k) represents the average k chosen over all the 33 datasets for each one of the distances. The sixth column (Max k) represents the maximal k with the respective distance, also over the 33 datasets.

In the seventh column, P-value is computed considering the accuracy values of each distance in relation to the accuracy obtained with Minkowski distance where p = 0.5, which was the distance that obtained the best mean accuracy (highlighted in bold in the second column of Table 3.2).

In summary, a total of 33 datasets from the UCI repository, 15 distances and values for k varying from 1 to 200 were evaluated. The proposed distance obtained accuracies that were better than the average more often than its counterparts (in 26 cases out of 33). Furthermore, it also obtained the best accuracy more frequently (in 9 out of 33 cases). As a remark, the proposed distance is also faster in than Manhattan and Euclidean in neighbourhood iterations.

Table 3.2: Average results obtained with 15 different distance measures (tested on 33 numerical datasets from the UCI repository).

Distances	Mean Acc	Mean TP	Mean TN	Mean k	Max k	P-Value	Better than Average	Best
Euclidean Distance	0.823	0.768	0.867	11.2	49	0.552	19	6
Chebyshev Distance	0.786	0.733	0.860	14.4	108	**0.129**	9	4
Manhattan Distance	0.834	0.780	0.878	10.7	70	0.787	24	4
Minkowski Distance (p = 0.5)	**0.842**	**0.789**	**0.884**	12.3	97	1.000	25	**9**
Minkowski Distance (p = 0.75)	0.837	0.785	0.882	10.8	68	0.874	24	4
Minkowski Distance (p = 3)	0.821	0.766	0.867	10.2	47	0.516	16	6
Minkowski Distance (p = 4)	0.820	0.766	0.867	10.5	47	0.490	18	8
Canberra Distance	0.695	0.627	0.828	24.1	190	**0.004**	10	8
Sum of Squared Difference	0.823	0.768	0.867	11.2	49	0.552	19	6
Proposed (Rodrigues) Distance (p = 0.5)	0.840	0.788	**0.884**	10.8	68	0.950	**26**	5
Proposed (Rodrigues) Distance (p = 0.75)	0.835	0.781	0.879	10.8	66	0.816	**26**	3
Proposed (Rodrigues) Distance (p = 1)	0.829	0.776	0.875	11.7	101	0.677	23	**9**
Proposed (Rodrigues) Distance (p = 2)	0.821	0.766	0.867	11.4	45	0.519	17	4
Proposed (Rodrigues) Distance (p = 3)	0.820	0.768	0.868	10.9	47	0.501	18	5
Proposed (Rodrigues) Distance (p = 4)	0.820	0.765	0.867	10.8	47	0.489	16	8

3.2.4 Complexity Analysis of MDC and MkNN

Theorem 3.2.3 provides the worst-case complexity of the proposed classifiers assuming that they use combinations of 2-dimensional models to generate a p-dimensional model.

Theorem 3.2.3 *The worst-case step complexity of MkNN and MDC is $O(W+H)$ for 2-dimensions and $O(p^2(W+H))$ for p-dimensional combinations of 2-dimensional classification models in the GPU, where W and H represent the width and height of the image, respectively.*

Proof *The neighbourhood of each instance in all implementations is checked respecting distance $d_{w_1,w_2,p}$ (see Section 3.2.3). Assuming that the classifiers run in the GPU, each pixel/element of the image is theoretically processed in parallel.*

The worst case scenario consists of growing element a until it covers the entire image. That is, the growth is equal to the diagonal length of the image (i.e., the distance between the element of the upper-left and the element of the bottom-right):

$$\begin{array}{rcl} O(a) & = & \\ O(d_{w_1,w_2,p}((0,0),(W-1,H-1))) & = & \\ O(w_1 \sqrt[p]{\sum_{i=1}^{n}|x_i-y_i|^p} + w_2 \max_{i=1}^{n}|x_i-y_i|) & & \end{array} \tag{3.28}$$

As $w_2 \max_{i=1}^{n}|x_i-y_i|$ is always equal or greater than $w_1 \sqrt[p]{\sum_{i=1}^{n}|x_i-y_i|^p}$, then:

$$\begin{array}{rcl} O(a) & = & \\ O(w_2 \max_{i=1}^{n}|x_i-y_i|) & = & \\ O(\max(W-1,H-1)) & = & \\ O(W+H) & & \end{array} \tag{3.29}$$

Therefore, the worst-case complexity is $O(W+H)$. In the ensemble approach, where 2D dimensional models are combined considering p attributes, its complexity is given by:

$$O(\binom{p}{2}(W+H)) \tag{3.30}$$

which is the cost for generating a 2D model multiplied by the number of combinations of every two attributes. Expanding this equation:

$$O(\frac{p!(W+H)}{(p-2)!})$$
$$O(\frac{p(p-1)(p-2)!(W+H)}{(p-2)!})$$
$$O(p(p-1)(W+H))$$
$$O((p^2-p)(W+H))$$

and finally:

$$O(p^2(W+H)) \tag{3.31}$$

■

Combining 2D models is much better in practice than computing dilations for p-dimensions directly. If we apply the expansions in 3 dimensions, for instance, we get a cubic complexity of $O(W^3+H^3+D^3)$, where W, H and D represent the width, height and depth. Let us suppose a hypothetical but frequent scenario such as: $W = H = D = sp$, where s is a scaling factor that relates the size of the grid to the number of dimensions of the problem. Then, with the proposed ensemble approach we have:

$$O(p^2(W+H)) = p^2((sp)+(sp)) = 2p^3s \tag{3.32}$$

operations. In the case of expanding the set partitions in p-dimensions directly, we have a total of:

$$O(W^3+H^3+D^3) = (sp)^3+(sp)^3+(sp)^3 = 3s^3p^3 \tag{3.33}$$

operations. Relating these two equations:

$$3s^3p^3 < 2p^3s \tag{3.34}$$

assuming that p and s are greater than 0, this inequality is only satisfied if $p > 0$ and $s < \sqrt{\frac{2}{3}}$, or, similarly, if $3s^2 < 2$. That is, the 2D combination approach will only achieve a greater number of steps if $s < \sqrt{\frac{2}{3}}$. However, s is greater than $\sqrt{\frac{2}{3}}$ for virtually every practical classification problem.

For instance, if the number of dimensions p of the problem is equal to 1, which is theoretically the best case, the complexity in the combinations case is $2s$ and in the p-dimensional case $3s^3$. In the best possible case it is already clear ($2s < 3s^3$) that it is more efficient to use 2D combinations instead of the direct p-dimensional approach.

3.2.5 2D Combinations in Practice

Let us suppose that an arbitrary input dataset has 4 dimensions, excluding the class label, such as the original UCI Iris dataset. In this case, we have a total of 6 combinations of 2 pairs of features (0 and 1, 0 and 2, 0 and 3, 1 and 2, 1 and 3, and 2 and 3). Each one of these combinations is a 2D dataset. 2-dimensional models are trained with each one of these combinations, and hence each combination produces a respective classification or predictive model. In total, we have 6, 2-dimensional classification models.

Now, let us suppose that we want to classify instances using these generated predictive models. Each combination produces a vote for the label of an unlabelled instance. These votes can be combined in a number of different ways. A simple approach would be to label unlabelled instances as the label that received more votes among the models. This is exactly what is done in the p-dimensional experiments in Section 4.1, although these votes could be combined in more complex ways.

We consider the fact that some of these 2D models do not produce very accurate votes. Thus, just the votes of the best 2D models are considered, i.e., the ones that achieved the best accuracies while training and testing with one of the 2D combinations (the training and validation is performed using 10-fold cross validation). The number of 2D models that should have their vote considered was chosen empirically for each experiment. We can fairly say, however, after all the performed experiments, that numbers close to 0 work far better than numbers close to $\binom{p}{2}$, as models with worst accuracies usually rely on features that are not very accurate (they do not split the data very well among classes).

It is clear that the number of combinations stems from the number of features of the dataset, which is not related at all to its number of instances. Iris has 4 features and 150 instances. Thus, the increment in the complexity using the combinations would be a vote (which is O(1), since for 2D models it is possible to use matrices, and one position in the matrix is visited), multiplied by the number of combinations. The complexity for generating a 2D model is $O(W + H)$, as previously shown. Thus, the worst-case complexity required for generating 6 of these models is given by the multiplication of this value with the number of combinations, which is given by $\binom{p}{2}$, where p is the number of

dimensions (4 for the Iris dataset). The resultant complexity is $O(p^2(W+H))$, which is the worst-case complexity reported in Theorem 3.2.3.

3.2.6 2D Combinations Theory

Multi-label p-dimensional classifications are defined as the aggregation of 2 dimensional binary classification models. A multi-label classification where class labels l belong to $\{1, ..., L\}$ is decomposed in L binary problems. For binary problem $l = 1$, the classification models predict if an unlabelled instance belongs or not to class 1. For binary problem 2, the same follows, i.e., the models are trained to predict if an unlabelled instance belongs to class 2 or not. This is repeated for all L class labels.

The final class label is given by the aggregation of these L votes, respecting the accuracy of each binary problem that is obtained using 10-fold cross validation during the training phase of the algorithm. Votes of classes that obtain the highest accuracies are computed first. Let us suppose we have the following set:

$$\begin{aligned} &Accuracy : 90.4\% \leftarrow label : 1 \\ &Accuracy : 70.1\% \leftarrow label : 2 \\ &Accuracy : 92.6\% \leftarrow label : 3 \end{aligned} \tag{3.35}$$

As class 3 has the highest accuracy, i.e., 92.6%, its votes are considered first in detriment of the others. That is, if this classification model classifies an unlabelled instance with label 3, then it is left as that. If this is not the case, i.e., not label 3, then the instance is tested for the model with the next higher accuracy, which is 90.4%, and the same procedure is repeated. If it is labelled as 1, then it is said to be label 1. Otherwise, it is labelled as 2. For this procedure to work properly, true positive rates are expected to be greater than true negative rates in the binary models. Otherwise, a contrary logic should be used instead. Logical combinations or even decision tables can be used as well for computing the final label.

A total of $\binom{p}{2}$ or less votes are considered for each binary labelling procedure. Some votes can be disregarded according to the accuracy of each 2D model, since they can provide weak or bad votes. At first, we rank all 2D models according to the accuracy obtained during the 10-fold cross validation in the training phase. Later, we consider a threshold parameter that indicates how many models should be considered in the voting. Say, if we have 22, 2D models (22 combinations of pairwise features), and the threshold

says that just the 10 best models are worthful, then the 12 least accurate models are disregarded. In this case, for the given binary problem, the algorithm will provide 10 votes, each one indicating if the instance belongs to a class or not. The majority of the votes indicates the predicted label.

3.2.7 Classification Model Compression

The classification models produced by our approach are, in theory, sets that contain p-dimensional instances. The storage of the model in practice can be performed using the said sets or using p-dimensional matrices.

3.2.7.1 Rectangular Compression for Sets

In the first case, the classification model is given by:

$$A_l^{I_l} = \{(h_1, ..., h_p)_1, ..., (h_1, ..., h_p)_n\} \tag{3.36}$$

where p represents the dimension of the problem, n represents the total amount of instances in the classification model and h represents the coordinate of each element.

For p dimensions, pnk bits are required to store the model as a set, as demonstrated below:

$$\begin{array}{rcc} \{(h_1)_1, ..., (h_1)_n\} & : & nk\ bits \\ \{(h_1, h_2)_1, ..., (h_1, h_2)_n\} & : & 2nk\ bits \\ \vdots & & \vdots \\ \{(h_1, ..., h_p)_1, ..., (h_1, ..., h_p)_n\} & : & pnk\ bits \end{array} \tag{3.37}$$

where k is the amount of bits designed for each variable that, in turn, represent coordinates of the elements in the set.

However, if rectangles are stored in a compressed notation (e.g., $(x : x + \theta_x, y : y + \theta_y)$ for two dimensions), where θ_j represents the size or range of the rectangle in dimension j, k extra bits are required to indicate this extended notation and further pk bits are required to store all θ_j. The required amount of bits is given by:

$$\begin{array}{rcl} \{(h_1 : h_1 + \theta_1, h_2 : h_2 + \theta_2)_1, & & \\ (h_1 : h_1 + \theta_1, h_2 : h_2 + \theta_2)_2\} & : & 2(4k + k)\ bits \\ \vdots & & \vdots \\ \{(h_1 : h_1 + \theta_1, ..., h_p : h_p + \theta_p)_1, ..., & & \\ (h_1 : h_1 + \theta_1, ..., h_p : h_p + \theta_p)_r\} & : & r(2pk + k)\ bits \end{array} \tag{3.38}$$

where r represents the total amount of rectangles.

A single rectangle of size $(\theta_1, ..., \theta_p)$ requires $(2pk + k)$ bits. Thus, $r(2pk + k)$ bits are required to store r rectangles. In contrast, storing the same r rectangles using Equation 3.37, assuming the rectangles are p-dimensional squares, requires $r(\theta_1 \times \theta_2 \times ... \times \theta_p)$ elements, which is the area of the p-dimensional rectangles multiplied by their quantity. This results to a $r(\theta_1 \times \theta_2 \times ... \times \theta_p)k$ bits requirement. The equivalence of these two equations is given by:

$$r(2pk + k) = r(\theta_1 \times \theta_2 \times ... \times \theta_p)k \tag{3.39}$$

and it is reached when p is greater than $(\theta_1 \times \theta_2 \times ... \times \theta_p)/2 - 1/2$. This shows that if p is lesser than $(\theta_1 \times \theta_2 \times ... \times \theta_p)/2 - 1/2$, the compression approach is much more efficient, which is virtually always the case (except for very, very small rectangles).

3.2.7.2 Partition Trees for Matrices

The second option is to consider images or matrices to store the classification models. In this case, another type of compression can be used. We propose the use of 2^p-trees to separate the matrix in p-dimensional quadrants, i.e., orthants. One of the positive aspects of working with matrices is that most implemented morphological operators work with matrices or, for the 2D case, images. Figure 3.7 shows a 2^p-tree representing a 4x4 binary image. Figure 3.8 shows the same 2^p-tree in a flat perspective, where the used orientations are: left-right and top-bottom.

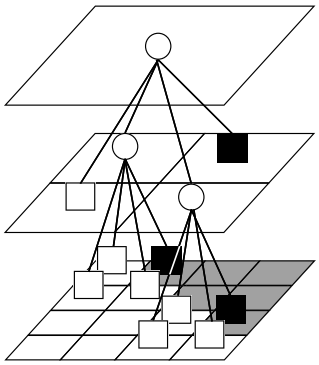


Figure 3.7: A 2D binary example of a 2^p-tree (a quad-tree in this case).

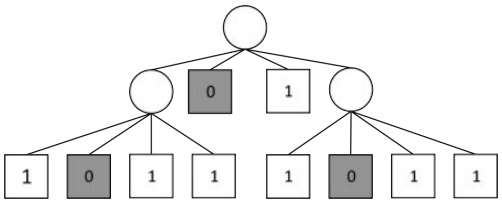


Figure 3.8: Flat representation of the 2^p-tree (a quad-tree in this case).

The 2^p-tree representation consists of dividing the entire matrix in 2^p orthants, recursively, until the desired representation precision has been reached. In Figure 3.7, the entire image is divided in 4 quadrants at first. The top-right and bottom-left quadrants contain uniform colors, black and white, respectively, and these nodes are promptly turned into leaves. The top-left and bottom-right quadrants are divided in 4 quadrants each, due to the fact that the pixel values in these quadrants are not uniform. The next level of the tree is the deepest level in this case, where the quadrants cannot be divided further. The leaves of this tree contain values for the binary case (black and white), but it is possible to generalize the representation for grey-scale images, i.e., multi-label classification models, where the leaves of the tree store the respective grey-level (or label) value.

It is clear that when the tree branches halt prematurely, storage compression is achieved. In Figure 3.7, two of the nodes of the second level are actually leaves, and were prematurely pruned due to the uniform pattern of the data in their respective quadrants. Thus, the 8 elements of the original image are compressed in 2. The worst-case complexity for reaching any element in the tree is $O(\log_{2^p} n)$, which is very efficient. The image shown in Figure 3.7 is written as:

- "XX01X10111011"

in depth-first order, which contains 13 values. 'X' represents a branch or parent node. The original image contains 16 values and the compressed version contains 13. In this case, a compression of 3 elements is obtained. A more significant compression can be observed in larger uniform classification models. Classification models obtained with the proposed algorithms, as shown in the results section, are highly prone to this type of compression, i.e., they are very uniform in most regions.

For p-dimensional classification matrices, each element in the string formulation must represent all possible classes in the problem, i.e., k must respect $2^k \geq L$, where L is the amount of possible distinct classes [99].

3.2.7.3 Run Length Encoding

Finally, a text-compression such as run-length encoding [40, 79, 83] can be applied to both approaches, i.e., it can be applied to the approach that uses sets as well as to the p-dimensional matrices approach, before storing the classification model. The set, as well as the 2^p-trees, can be written as a single string and therefore text-compression and run-length encoding apply.

Run-length encoding consists of replacing characters repetitions with a unique occurrence of the character along with the times that the occurrence happens in sequence. The sequence "AAAABBC" can be written as "3A2BC", which requires less memory. Run-length encoding can also be applied along with some pattern recognition approaches, where patterns that occur the most can be substituted by another predefined set of patterns that consume less memory [17].

3.2.8 Parameter Selection

In this section, we will shift our attention to parameter optimization. The performance of classifiers depends on the chosen set of input parameters. In order to fairly compare the performance of the morphological classifiers to other classification algorithms and to visualize the obtained results, we performed two separate sets of experiments. The first set consists of experiments with 2 dimensional datasets. That is, two p-dimensional datasets were reduced to 2 dimensions for better visualization, and the parameters of each classifiers were tuned using an evolutionary algorithm (EA) [88].

The parameters were optimized using two methodologies. The first consists of using EA for tuning of numerical parameters, while categorical attributes were manually con-

figured. The second approach uses Auto-Weka [103], which determines the best suited algorithm from the Weka [36] machine learning framework and an appropriate set of parameters, including a subset of the dataset attributes if it leads to a performance improvement.

3.2.8.1 Evolutionary Algorithm

The proposed EA consists of a combination of an elitist strategy, mutation and cross-over. The size of the population carried on to the next generation was set to 20. At first, 20 individuals are randomly generated. In each generation, a single individual is created. This individual receives randomly selected parameters of two randomly selected parents (cross-over). After that, mutation is performed on the selected parameters. This individual has its objective function computed to confirm whether it may enter in the top 20. If that is true, then the individual with the smallest objective function value among the top 20 is removed. The objective function is defined as the accuracy of the classifier on 10-fold cross validation. Algorithm 7 illustrates the overall process of the EA.

Algorithm 7: Evolutionary algorithm to select the numerical set of parameters for the classification algorithms.

Data: Θ is the population at each generation, $\Psi()$ is a function that generates a random number $\in [0, 1]$, ε represents individuals and Π represents a maximal value of each parameter (a predefined threshold), chosen according to the classification algorithm.

```
1  begin
2      w ← 0;
3      Θ ← randomly generate 20 individuals;
4      while the target time has not been reached do
5          ε1 ← take the (40Ψ()%20)^th best fit individual in Θ;
6          ε2 ← take the (40Ψ()%20)^th best fit individual in Θ;
7          ε_new ← randomly fuse the parameters of ε1 and ε2 (e.g., take parameter 1
             from ε2, parameter 2 from ε1, parameter 3 from ε1, etc, where this choice
             is completely at random until all parameters are populated);
8          ε_new ← perform a mutation on ε_new if Ψ() < 0.6, i.e., for each parameter s,
             if Ψ() < 0.6, then the parameter s = s + (Ψ()Π_s − Ψ()Π_s/2) / (w/50);
9          if the objective function of ε_new is higher than of any individual in Θ then
10             Take out the least fit individual from Θ and insert ε_new;
11         w ← w + 1;
```

Each EA run was limited to a total of 12 hours of evolution. This was also the case for Auto-Weka. Since MkNN and MDC were faster than the algorithms in Weka, for these

cases, the genetic EA was set to converge in 6 hours.

3.3 Clustering

As the methodology that concerns to classification has been presented, we will now shift our attention to the clustering proposal. As opposed to the previous section, where two classification algorithms were proposed, this section focuses on proposing a single clusterization algorithm, called k-Morphological Sets (k-MS), which is capable of segmenting and grouping data respecting shapes and cluster density [89]. The method is mostly based on morphological reconstruction.

Before heading to how the algorithm works, we define how data is preprocessed. At first, an image or grid G is constructed. $G(x, y)$ returns the index value $x \times y$ in case there is an instance at position (x, y) or 0 otherwise. An image mask $M(x, y)$ returns 1 in case an instance is positioned at (x, y) and 0 otherwise.

Once image G is created, a reconstruction $R_M^B(G)$ is applied. The structuring element B in Figure 3.9 is used, where its element values are 0 (including the central element marked with $\times$) and the total size is 1 with respect to the *sup* metric. The size of the structuring element can be altered, which leads to distinct types of clusterizations. The shape of the element can be modified as long as it contains all directions (e.g., left, right, up and down for the case of 2 dimensions). If this is not the case, the algorithm may reach a potential deadlock.

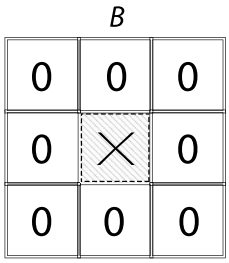


Figure 3.9: Shape and values of the structuring element B.

A structuring element with contain only zeros implies that the dilation would just spread the biggest values through some parts of G (limited by mask M), biased by the size and shape of the structuring element. Therefore, after the reconstruction, a clusterized image is obtained, such as in Figure 3.10. Figure 3.10-(a) illustrates a possible image G with grey values $\{1, 2, 8, 9, 12, 14, 17, 18\}$ (which are also the indexes of the pixels), all the

other pixels are set to background (and are not shown in the figure). Figures 3.10-(a) to (e) illustrate the morphological reconstruction as time advances. In Figure 3.10-(e), we have two clusters after a total of 4 iterations (a to d) and idempotence has been reached.

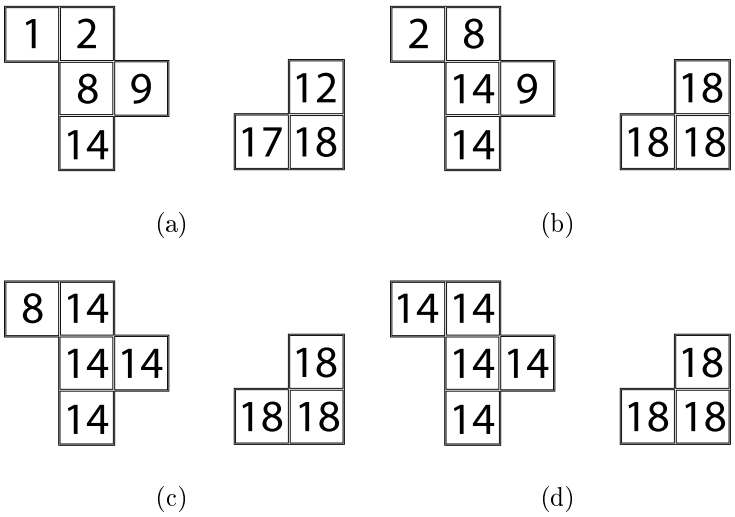


Figure 3.10: An indexed grey scale morphological reconstruction over a number of iterations.

This morphological reconstruction does not separate input data in a prescribed number of clusters. Instead, the algorithm separates the data based on "connectivity", relying on the size and shape of the structuring element. Therefore, we introduce an internal variable δ that is responsible for increasing the size of the structuring element B. That is, in the beginning of the clusterization, the size of δ would be minimum, as depicted in Figure 3.11-(a). With a structuring element of this size, the algorithm reaches an idempotent configuration. Therefore, we gradually increase the size of the structuring element until it reaches the remaining grey instances at the bottom of the image, shown in (b). At each increment of δ, the algorithm checks for the condition of idempotence, and if it is still idempotent, it increments δ. Otherwise, the algorithm resets δ to the minimum, such as shown in (c). At that stage, the grey level value of the instances at the bottom reaches the upper instances, passing the index of the bottom cluster to them (d). Figure 3.11 depicts a hypothetical situation where $k = 1$, i.e., just one cluster is desired.

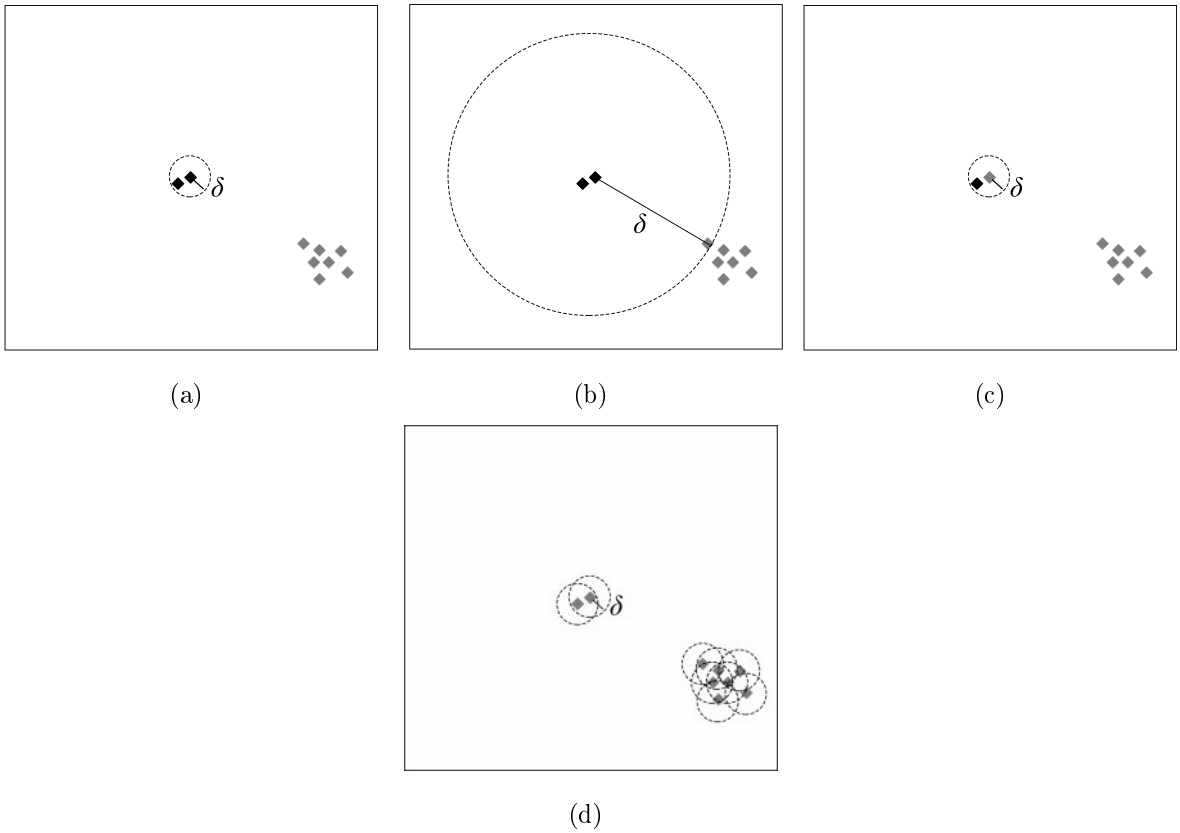


Figure 3.11: Some steps of the proposed k-MS algorithm.

Algorithm 8 shows a high-level overview of the proposed algorithm, where δ is a factor that multiplies the size of the structuring element B. In the beginning of the algorithm, δ is set to 1. A morphological reconstruction operation is then performed on image G using the structuring element B and mask M. The reconstruction consists of dilating the image and applying the mask until idempotence has been reached, as exemplified in Section 2.1 and also in Figure 3.10.

Later, the number of unique values in G is computed, which is equivalent to the number of clusters present in G at that moment. This computation is performed using early breaks, as shown in Algorithm 9, to speed up the processing. If G has more than k clusters, the algorithm continues to the next step, where δ is incremented. G is dilated once. If G is idempotent, then δ is incremented until G is not idempotent. The idempotence is overcome when an index that belongs to a certain cluster reaches another with a different index. When this is the case, the algorithm resets δ to 1 and starts the same process all over again. With less than k or exactly k clusters in G, the algorithm terminates. In contrast to the more general view, Algorithm 9 shows a more detailed, low level version of the k-MS algorithm.

Algorithm 8: High-level k-Morphological Sets algorithm.

Data: G is the grey scale image that contains xy a value at position (x, y) if there is an instance in the input dataset in this position, and 0 otherwise. M is the mask previously described, B the structuring element, k is the desired maximal number of clusters to be created and R the reconstruction function

1 **begin**

2 $\delta \leftarrow 1$;

3 Perform reconstruction $R_M^{\delta B}(G)$;

4 If the number of clusters in G is less or equal to k, i.e., if there is a total number of k or less different grey level values in G, stop the algorithm here;

5 $\delta \leftarrow \delta + 1$;

6 Dilate G, that is: $G \leftarrow dil^{\delta B}(G)$;

7 If G is idempotent ($G = dil^{\delta B}(G)$) then return to line 5;

8 Otherwise, return to line 2;

Algorithm 9: Low-level k-Morphological Sets algorithm.

Data: B is the structuring element, k is the desired maximum number of clusters to be created and R the reconstruction function

```
1  begin
2      finished ← false;
3      δ ← 1;
4      while !finished do
5          idempotent ← true; finished ← true;
6          Reset or remove values from the kArray (the array contains at most k
            different values);
7          for every instance or data point p in the input dataset do
8              p_aux ← p;
9              If the surroundings of pixel p, respecting the structuring element B and
                δ, contains a higher value than p, then p receives the highest
                surrounding value;
10             if !idempotent then continue;                  // first early break
11             if p_aux ≠ p then idempotent ← false;
12             if !finished then continue;                    // second early break
13             if kArray does not contain the value of p then
14                 if kArray is fully populated then finished ← false;
15                 else Add the value of p to kArray at a vacant position;
16         if idempotent then δ ← δ + 1;
17         else δ ← 1;
18         finished ← finished and idempotent;
19     Returns all the instances p where each one is indexed with a unique cluster
        index (amount of clusters ≤ k);
```

3.3.1 Complexity Analysis of k-MS

As it can be seen in Algorithm 9, an inner "for" loop depends on the total number of instances n in the dataset. Within this loop, there is another loop that is dependent on the k variable. Thus, in the worst case scenario, a total of $O(n \times k)$ steps is required for computing a single dilation of the dataset. The maximum number of dilations for two clusters to reach each other in the worst case is equal to the distances of the two farthest instances in the dataset divided by the initial δ (which is equal to 1 in our case).

Let us assume that $\hat{d}$ represents this maximum distance between the two farthest instances in the dataset. Then $\hat{d}/\delta$ represents the maximum number of dilations that must be performed for two clusters to reach each other. Since k is the number of desired clusters and assuming the worst case scenario, a single cluster is produced by the algorithm regardless of $k > 1$ (this would not happen in practice, however it serves for the purposes of complexity analysis), then $\hat{d}/\delta$ would have to be performed a total of k times at worst case. This leaves us with the complexity shown in Equation 3.40, which is equivalent to $(k\hat{d}/\delta) \times (nk)$.

$$O\left(\frac{k^2\hat{d}n}{\delta}\right) \tag{3.40}$$

Variables δ and k are constants and independent of n so they can be excluded from the analysis. Thus, this results in a $O(\hat{d}n)$ complexity. The variable $\hat{d}$ can be expressed in terms of n because if the number of instances in the dataset increases, the maximum distance between two possible instances can also increase. In the worse case scenario we would have $O(n^2)$.

As a remark, Karypis et al. [44] do not provide a complexity or time analysis regarding Chameleon in their work. D. Liu et al. [52] and Yousri et al. [114] provide a complexity of $O(n\,log\,n)$ for the TRICLUST and Mitosis algorithms. Although the worst case complexities of their algorithms are better than ours, it is possible to argue that in practice our algorithm was faster, as discussed in the following experiments and conclusion chapters. This also reinforces the fact that computational complexity is different to run time analysis.

3.3.2 Parallel and GPU Aspects

Besides the CPU implementation of the algorithm, we propose and evaluate parallel implementations of k-MS, which includes a GPU implementation. The GPU algorithm is divided in two kernels. The first kernel receives a binary image M containing input instances. $M(x, y)$ equals 1 if an instance belongs to that position and equals 0 otherwise. G is constructed based on M, and contains unique indices for each instance. Figure 3.12 illustrates the steps of the k-MS algorithm and its synchronization with the CPU.

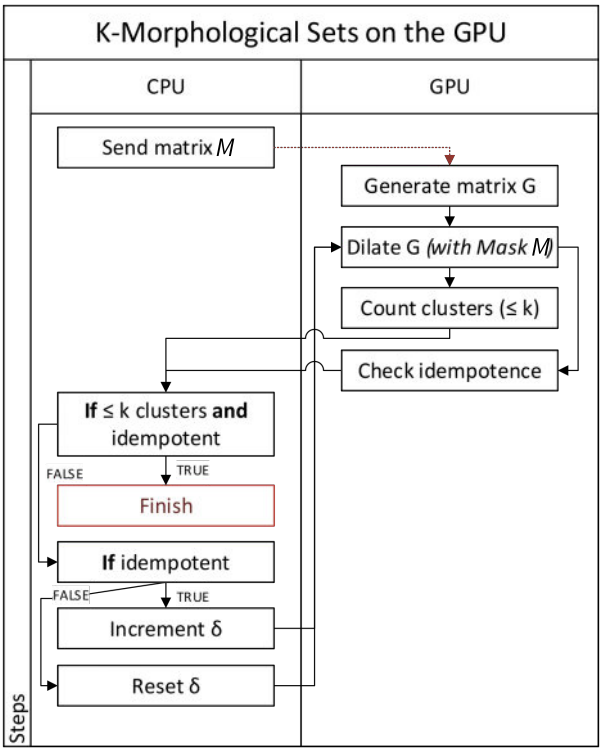


Figure 3.12: Overall steps of the k-MS GPU implementation.

The parallel versions of the algorithm (CPU and GPU) work very similarly to the one shown in Algorithm 9. The main difference is that the "for" loop in line 7 is processed in parallel. In the GPU implementation, the early breaks in lines 10 and 12 do not produce an improvement of processing times, since GPUs use the SIMD (Single Instruction Multiple Data) paradigm, where the same instruction is performed in every thread at the same time. Thus, even if some threads escape from the loop early, other threads that are scheduled to run together will eventually push the processing time to the worst case, as if the early break instructions are not present.

However, the early breaks do obtain improvements with CPU parallel implementations because they do not adhere to the SIMD paradigm. In fact, they are able to process different instructions in parallel. Furthermore, the performance of the GPU algorithm will not be very good for large values of k. This is true because of two facts: (1) the early breaks do not work, and (2) more atomic operations have to be performed in each position of the k-array to avoid concurrency of the various threads, and performing a slow operation several times adds a significant overhead to running times.

The cluster counting in the GPU is performed as follows: after the dilation of each pixel in G (line 9 of Algorithm 9), a "for" loop iterates through an array of size k. If the value is not within the array and the array is not full, then the value is placed at a

vacant position using the *atomic compare and swap* operation. Otherwise, if the value is not within the array and the array is full, then an *atomic and* sets the *finished* boolean variable to false. Similarly to the sequential version, the output of the algorithm is the G matrix with k or less distinct indexes or grey level values (ignoring the background value) and the k-array, containing every index of every cluster in the matrix. Each cluster has a unique and distinct index with relation to the remaining ones.

3.3.3 Memory Aspects

The k-MS algorithm requires $g(n, L, k)$ bits of memory, as shown in Equation 3.41, where n represents the number of dimensions of the instances, L represents the amount of instances and k stands for the size of the k-array, which is also the number of desired clusters. Integers of 32 bits were used in the implementation. However, this can be increased or decreased as desired, depending on the input dataset.

$$\begin{aligned} g(n, L, k) = sizeof(int) \times (nL + k) + sizeof(\delta) \\ + sizeof(finished) + sizeof(idempotent) \end{aligned} \tag{3.41}$$

In our specific case, we set δ to an integer so that $sizeof(\delta)$ is equal to 32 bits and $sizeof(finished)+sizeof(idempotent)$ equals to 2 bits, each one being Boolean variables. The experiments in this work were not sufficient to overflow the memory of the used GPU (Nvidia GTX 960M). In addition, the algorithm requires subtle amounts of local memory, which would not overflow any GPU shared nor local memory.

3.4 Summary

In this chapter, we provided definitions, theoretical frameworks and algorithmic implementations for classification and clustering. Proposed classifiers are based on a neighbourhood iteration paradigm, where a novel efficient distance is used. The clusterization algorithm, on the other hand, does not use this distance.

An extensive complexity analysis was performed for both approaches. In the classification case, this analysis is used to prove that combining 2D models is in fact more efficient than solving problems using p-dimensional approaches in the worst case. A formal compression method is also proposed at the end of the classifiers section.

The clusterization algorithm, on the other hand, is simpler and is currently tuned towards image processing. However, it can be readily applied to p-dimensional datasets. Morphological operations support p-dimensions, where, in this case, each element in the set is a p-dimensional element. The elements of the structuring element are also p-dimensional.

A complexity analysis and GPU implementation regarding the clusterization method were presented. In this case, we discussed how the GPU SIMD paradigm impacts on performance. In the next chapter, we perform experiments comparing both implementations to state-of-the-art algorithms.

Chapter 4

Machine Learning Experiments

This chapter is segmented in two main sections and comprises all the experiments performed in this thesis with the exception of the experiments related to the proposed distance, which were presented in the previous chapter. The first section (4.1) addresses the experimental results and comparisons regarding the classification approach. The second section (4.2) addresses the experimental results of the clusterization method and its comparison to state-of-the-art algorithms.

4.1 Classification

First and foremost, we introduce the framework used in this section for the classification experiments. Weka [36], which is a Java based machine learning framework, was the used framework, apart from MDC and MkNN. MkNN and MDC were programmed in CUDA/C/C++ and their source code is available at [81], along with a list of datasets. All the implementations proposed in this work conform to the use of the complement approach shown in Theorem 3.2.1 due to the described benefits.

The following subsection focus on comparing classifiers regarding 2-dimensional binary datasets. In this case, it is possible to visualize the results produced by the proposed morphological classifiers, and how they behave in each case. We briefly address the used datasets, and later provide the results obtained with this experiment. Later, a set of experiments that use multi-label p-dimensional datasets is addressed in Section 4.1.2.

4.1.1 Visual 2-Dimensional Experiments

In this section, we describe the details of used 2D datasets as well as the performance of the classifiers, visual results and comparisons. In Section 4.1.2, experiments with p-dimensional datasets are presented.

The first implementation of the proposed classifiers considers input datasets as multisets. The second implementation excludes repetitions. The subscript *Rep* in the name of the algorithms indicates the version that uses the multiset concept. Combining 2D models is an efficient way to approach the problem using morphological classifiers, as demonstrated in Section 3.2.4.

4.1.1.1 Used 2D Datasets

Two reduced versions of datasets from the UC Irvine (UCI) repository [107] were considered in this set of experiments. The first one is the commonly known *Iris* dataset. The number of attributes of this dataset was reduced to 2. The attributes *sepallength* and *petallength* were randomly selected and the remaining were excluded, except for the class label. The instances whose classes were equal to *Iris-setosa* were also removed from the dataset in order to transform this problem into a binary classification problem.

The second dataset was *Diabetes*, and the selected attributes were *plas* and *insu*. The original dataset already consists of a binary classification problem and thus no instance was removed. Table 4.1 illustrates this information and the total number of instances of each dataset.

Table 4.1: Datasets used in the 2d experiments.

Dataset	Selected Attributes	Number of Instances
Iris	sepallength, petallength	100 (per label: 50 - 50)
Diabetes	plas, insu	768 (per label: 500 - 268)

4.1.1.2 Iris Dataset

The results obtained with the Iris dataset and each algorithm (each optimized with EA and manual adjustments) as well as the Auto-Weka approach are shown in Table 4.2.

Table 4.2: Accuracies in percentage obtained with the modified Iris dataset.

Classifier	Acc (%)	TP (%)	TN (%)
Multilayer Perceptron [70]	**95**	92	98
RBF Network [11]	94	**94**	94
Conjunctive Rule [70]	94	92	96
SPegasos [70]	93	92	94
k-NN (IBk) [4]	93	**94**	92
SVM [20]	93	92	94
SMO [73]	93	92	94
Auto-Weka [103]	93	92	94
REPTree [70]	92	88	96
Random Forest [10]	91	92	90
Decision Table [28]	91	88	94
C4.5 (J48Graft) [70]	91	88	94
C4.5 (J48) [77]	91	**94**	88
Bayes Net [13]	91	88	94
Naive Bayes [92]	91	88	94
Hoeffding Tree [41]	87	88	86
MkNN	**95**	**98**	92
MkNN$_{Rep}$	94	**96**	92
MDC	**95**	**100**	90
MDC$_{Rep}$	**95**	**98**	92

Rep stands for the version of the algorithm that regards repetitions (multiset).
The algorithms highlighted in **bold** are the ones being proposed in this work.

Apart from the morphological classifiers, the Multilayer Perceptron achieved the highest accuracy, 95%. Three implementations of the proposed algorithms also achieved 95% of accuracy with this dataset (MkNN, MDC and MDC$_{Rep}$). In 2-dimensional cases, MkNN and MDC produce high true positive rates, usually higher than other classification algorithms while maintaining high accuracies (this is more evident with the MDC algorithm).

Figure 4.1 shows the 2D Iris dataset. The classes that are equal to 1 are shown as red circles while the ones that are equal to 2 are depicted as green squares. In this case, the dataset is nearly linearly separable.

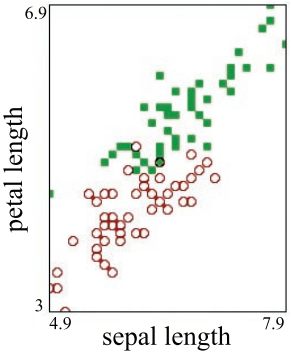
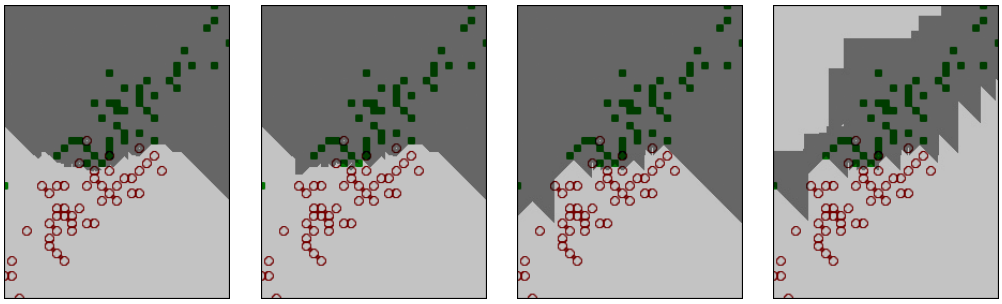


Figure 4.1: Used modified Iris dataset.

10-fold cross validation separates the dataset in 10 partitions. At the first iteration, a classification model is created trained on $10-1$ partitions and tested on the remaining 1. This process is repeated 9 more times while always switching the test partition to a partition that has not yet been used as a test set.

Figure 4.2 shows one of the classification models of the 10 iterations (or folds) of the Iris dataset shown in Figure 4.1. These images are classification models $A_1^{I_1}$ of the implementations (a) MkNN, (b) MkNN$_{Rep}$, (c) MDC and (d) MDC$_{Rep}$, respectively. The 10 generated classification models for each fold can be found in [81].



(a) MkNN (b) MkNN$_{Rep}$ (c) MDC (d) MDC$_{Rep}$

Figure 4.2: Generated classification models on Iris.

The darker grey area in Figure 4.2 represents what does not belong to $A_1^{I_1}$, while the lighter grey represents instances that belong to $A_1^{I_1}$. Assuming we are disregarding the repetition of instances and conceiving the fact that this dataset almost does not contain repeated instances, then the more the lighter gray area intersects instances labelled as 1 (red circles) and excludes the ones labelled as 2 (green squares), the higher the accuracy of the model. This statement is true in general for similar cases.

4.1.1.3 Diabetes Dataset

Figure 4.3 shows the 2D Diabetes dataset. Figure 4.4, on the other hand, shows the classification models of one fold of each proposed algorithm. In the case of MDC_{Rep}, the orientation parameter β chosen by the EA was equal to all directions but left. Therefore, the dilator clearly grows the structure in the other 3 directions. In the case of MDC, β restricted the growth downwards.

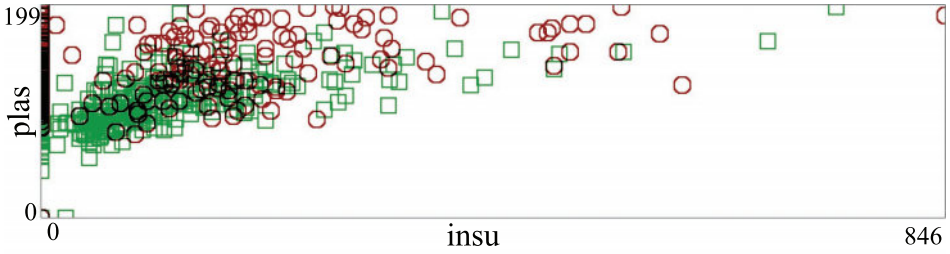
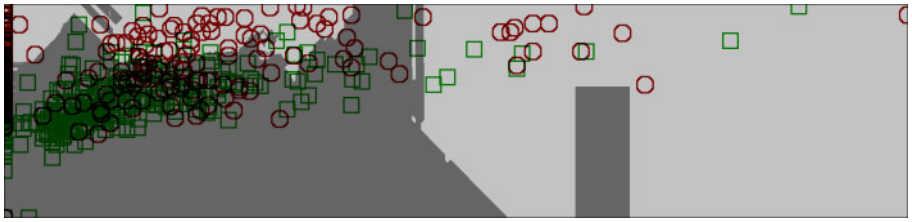
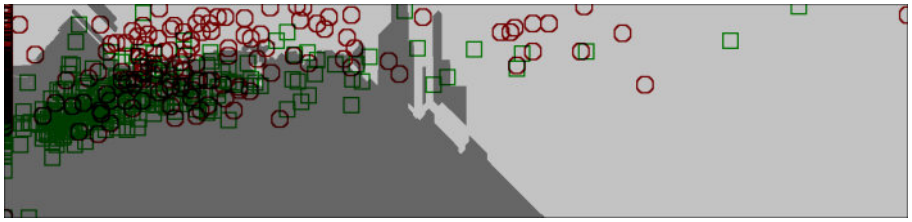


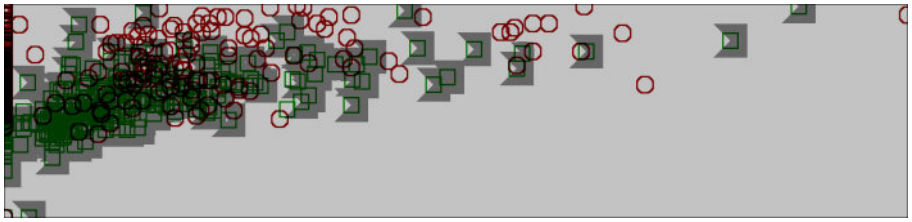
Figure 4.3: Used modified Diabetes dataset.



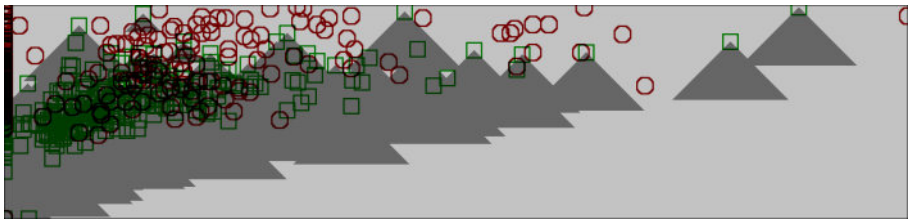
(a) MkNN



(b) MkNN$_{Rep}$



(c) MDC



(d) MDC$_{Rep}$

Figure 4.4: Generated classification models on Diabetes.

This dataset, in particular, has more repetitions than the previous Iris dataset, and therefore it is a difficult to infer something from the classification models shown in Figure 4.4.

Table 4.3 shows the obtained results on the 2D Diabetes dataset. In this case, MkNN$_{Rep}$ and SMO outperformed the remaining algorithms, and tied on the accuracy. However, MDC still produced a high true positive rate. Auto-Weka, in this case, also selected LMT as the best classifier.

This case was the worst for the proposed algorithms, where just for MkNN$_{Rep}$ case, all the averaged indexes of the classifiers in comparison have been outperformed.

Table 4.3: Accuracies in percentage obtained with the modified Diabetes dataset.

Classifier	Acc (%)	TP (%)	TN (%)
SMO [73]	**74.86**	**91.6**	43.7
SPegasos [70]	74.60	90.6	44.8
Multilayer Perceptron [70]	74.60	88.2	49.3
SVM [20]	74.60	89.8	46.3
Auto-Weka [103]	74.47	88	49.3
Hoeffding Tree [41]	74.08	89.4	45.5
RBF Network [11]	73.82	90.4	42.9
Naive Bayes [92]	73.69	89.8	43.7
k-NN (IBk) [4]	73.43	84.8	52.6
REPTree [70]	73.04	86.6	47.8
C4.5 (J48Graft) [70]	73.04	88.8	43.7
C4.5 (J48) [77]	73.04	88.8	43.7
Decision Table [28]	72.65	90.6	39.2
Conjunctive Rule [70]	72.52	81.4	56
Bayes Net [13]	71.35	83.6	48.5
Random Forest [10]	68.48	80.4	46.3
MkNN	72.65	88.6	46.2
MkNN$_{Rep}$	**74.86**	88.6	49.2
MDC	71.61	**91.4**	37.7
MDC$_{Rep}$	73.43	89	44.4

Rep stands for the version of the algorithm that regards repetitions (multiset).
The algorithms highlighted in **bold** are the ones being proposed in this work.

4.1.2 P-Dimensional Experiments

In this subsection, experiments and comparisons are performed with p-dimensional binary and multi-label datasets. We considered 8 datasets from the UCI public repository. The proposed p-dimensional approach uses a combination for 2-dimensional classification models, due to previously described advantages (see Theorem 3.2.3).

Each pairwise combination of features generates a binary predictive model. The total number of models is given by $\binom{p}{2}$, where p represents the number of features or dimensions of the dataset. An unlabelled instance is classified using the votes of some of the $\binom{p}{2}$ models, where each vote indicates if an instance belongs or not to a class label.

Multi-label datasets were separated in L binary problems, where L is the number of labels in the dataset. That is, binary classifiers were trained for each one of the L classes. Let us suppose a three-classes multi-label problem (classes 1, 2 and 3). For class 1, the instances in the dataset either belong or not to class 1. Thus, this constitutes a binary problem for class 1. Two other binary problems are created for class 2 and 3.

Next, for the binary problem of class 1, the approach follows as previously described. $\binom{p}{2}$ binary classification models are generated for the class 1 problem. Each model provides

a vote for an unlabelled p-dimensional instance. In total, we have $\binom{p}{2}$ or less (according to the significance of the model/vote) votes that decide if the instance belongs to class 1 or not. The procedure is repeated for the other two possible classes, class 2 and class 3. At the end, each binary problem (class 1, class 2 and class 3) provides a vote for each one of the class labels. The majority of votes indicates the predicted label. Section 3.2.6 addresses this decomposition logic in more detail.

Table 4.4 contains the accuracies obtained with each classifier for each dataset. The proposed classifiers obtained competitive accuracy rates. In the case of Iris, Tae, Haberman, Heart Statlog and Breast W, the proposed classifiers obtained the highest accuracy. This proves that using mathematical morphology in classification is feasible. Further enhancements, however, are still necessary for constructing a top-tier morphological classifier.

Table 4.4: Accuracies in percentage obtained with each dataset and classification algorithm.

Classifiers	Iris	Diabetes	Liver Disorders	Tae	Column 2C	Haberman	Heart Statlog	Breast W
RandomForest	95.33	**77.78**	61.15	68.87	81.48	69.28	81.48	96.70
Multilayer Perceptron	97.33	75.67	65.07	54.96	78.51	69.28	78.51	94.99
RBFNetwork	95.33	75.39	**66.95**	52.98	84.07	73.85	84.07	95.85
Conjunctive Rule	66.66	71.87	60.86	37.74	74.07	73.52	74.07	91.70
kNN	96.66	74.73	57.68	62.25	81.11	73.85	81.11	96.70
SVM	96.66	77.47	56.81	54.30	66.66	75.49	66.66	96.85
SMO	96.66	77.34	64.63	58.27	84.07	73.85	84.07	97.28
REPTree	94.00	75.26	61.44	54.30	78.51	73.52	78.51	94.27
DecisionTable	92.66	70.18	63.47	47.01	**84.81**	72.54	**84.81**	95.27
J48Graft	94.66	73.69	60.28	60.26	77.07	73.20	77.07	94.84
J48	96.00	73.82	59.13	59.60	76.66	72.87	76.66	94.56
NaiveBayes	96.00	76.30	64.05	54.30	83.70	**76.14**	83.70	97.13
BayesNet	94.00	74.34	61.44	47.01	81.11	72.54	81.11	97.13
HoeffdingTree	95.33	76.17	63.76	53.64	83.30	74.83	83.30	95.99
MDC$_{Rep}$	**98.00**	75.39	65.79	66.66	81.29	75.81	84.07	**97.42**
MkNN$_{Rep}$	**98.00**	75.13	66.37	**72.46**	81.93	**76.14**	**84.81**	96.99
MDC$_{Rep}\star$	4.62	0.39	3.88	11.98	1.64	2.61	4.42	1.76
MkNN$_{Rep}\star$	4.77	0.34	4.40	18.87	2.42	2.64	5.30	1.41

$\star$ represents the percentage improvement in comparison to the averaged accuracy of all classifiers.
MDC$_{Rep}$ and MkNN$_{Rep}$ represent the two proposed classifiers computing repetitions.

Table 4.5 shows the training and testing times of MDC and MkNN for each evaluated

dataset. Training times include the phase to train and build the 2D models for all the pairwise feature combinations as well as the multi-label training, if applicable for the dataset, considering 9/10 of the original dataset. Testing times include the time for testing the dataset in 1/10 of the dataset (one partition of the 10-fold cross validation). Although training requires a little bit of time, testing times are remarkably fast due to the nature of the classification model, which are matrices whose elements are accessed with O(1) complexity in the worst case.

Table 4.5: Training and testing times (s) obtained with MDC_{Rep} and MkNN_{Rep} in seconds in the GPU.

Datasets	MDC Train Time (s)	MDC Test Time (s)	MkNN Train Time (s)	MkNN Test Time (s)
Iris	0.1223382649	0.0000030028	0.1594423040	0.0000009341
Diabetes	6.4961881040	0.0004020527	14.2825142510	0.0002412080
Liver Disorders	1.8445822000	0.0000200820	0.9116863500	0.0000295500
Tae	3.6113764500	0.0000900310	3.6986199000	0.0006007500
Column 2C	2.9733453099	0.0000601520	6.1188321300	0.0000798789
Haberman	0.8608457292	0.0000100440	0.8368868598	0.0000100000
Heart Statlog	9.3438252208	0.0001833519	13.3958921399	0.0023547693
Breast W	5.1447441331	0.0003430842	13.9826192997	0.0003608940

Real test times may be even lesser than the ones reported in Table 4.5. The test-time is very fast due to the O(1) classification complexity for each 2D attribute combination. Therefore, reported times may be overly influenced by program initialization. In summary, both training and test times are feasible, although we may not see a significant improvement in processing times related to the GPU paradigm with these datasets.

4.2 Clustering

We will now shift our attention to the clusterization experiments. Three distinct groups of experiments are performed in this section. The first group, described in Section 4.2.1, evaluates and compares the results obtained with k-MS and other popular clusterization algorithms that are usually employed in high dimensional clusterization and do not necessarily aim to be sensitive to density and shapes. The results of the clusterizations are analysed by volunteers, who were instructed to select the most human-like segmentations.

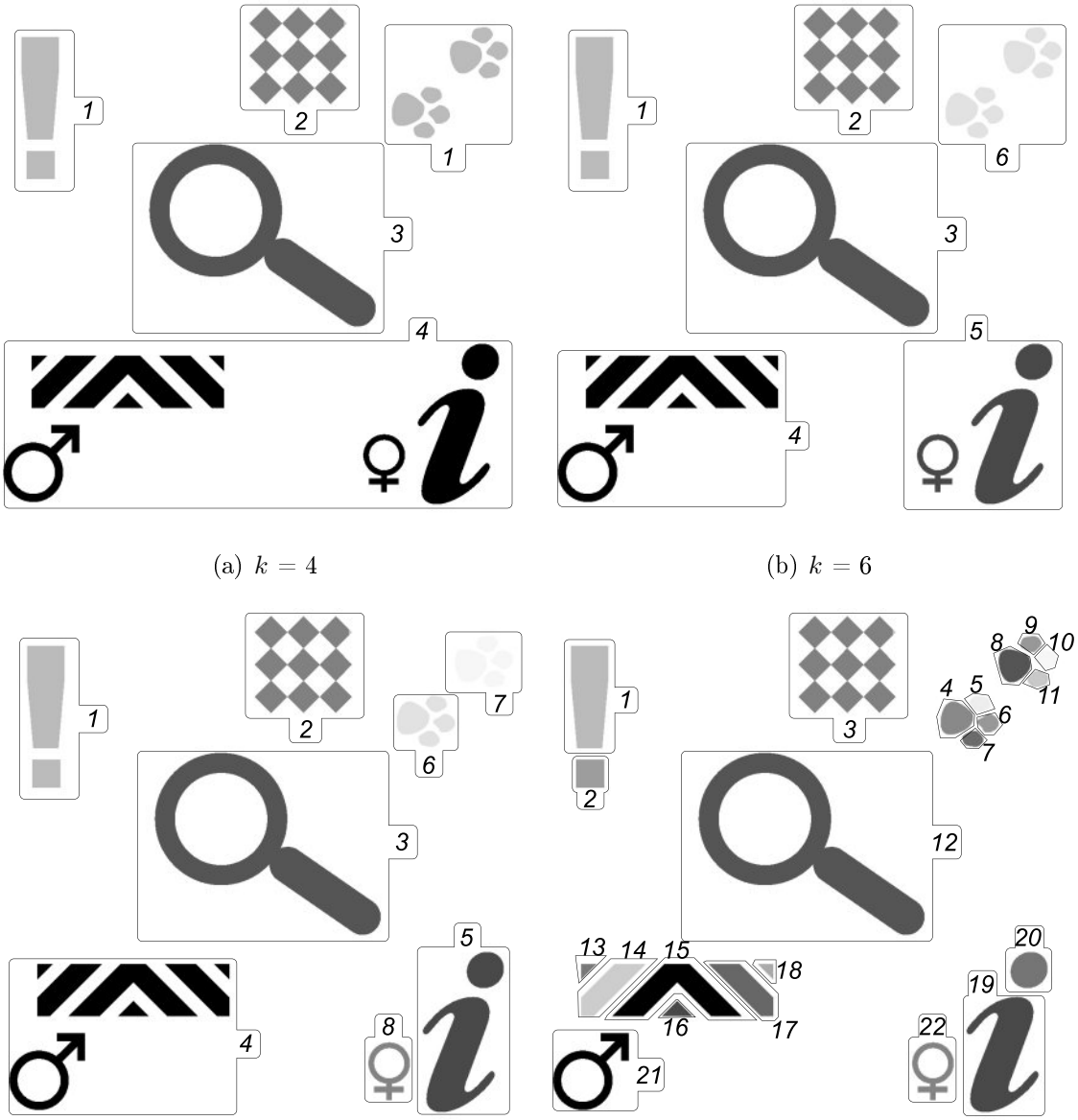
The visual output of k-MS algorithm is also analysed as variable k is increased. The dataset used in this section is noiseless.

The second group of experiments, addressed in Section 4.2.2, compares the visual results obtained in a noisy publicly available dataset [44]. In this case, we compare k-MS to clusterization algorithms with similar intents, i.e., algorithms that are sensitive to density and shapes. The behaviour of k-MS is also analysed using different input parameters. Finally, in the third group of experiments (Section 4.2.3), we perform an extensive time analysis, comparing the running times obtained by the different implementations of k-MS with the other clusterization algorithms. The structuring element used for almost all of this section is shown in Figure 3.9.

4.2.1 Noiseless Morphology Experiment

The input data shown in Figure 4.5 is considered for this noiseless group of experiments, where the black pixels represent the instances in a 2 dimensional space. The first experiment consists of running k-MS while varying the input k variable in order to visualize how the algorithm groups different objects in the image gradually. Figure 4.6 shows the results of the clusterizations for different values of k.

Figure 4.5: The input T matrix.



(a) $k = 4$ (b) $k = 6$

(c) $k = 8$ (d) $k = 25$

Figure 4.6: Visual results of k-MS with varying values of k.

It is possible to see in Figure 4.6-(a) that, for $k = 4$, k-MS grouped the exclamation mark and the paws in the same cluster. This occurs because the image boundary was disregarded. That is, if x coordinate being iterated end up being larger than the *width* of the image then the $x\%width$ position is considered instead. For $k = 6$, in image (b), the algorithm separated this cluster and the big cluster at the bottom that were previously grouped together for $k = 4$. The amount of valid clusters keeps increasing until $k = 22$. If $k > 22$, the algorithm still separates the dataset in 22 clusters, such as shown in (d), where k is equal to 25.

In addition, we compare the clusterization results obtained by k-MS considering two

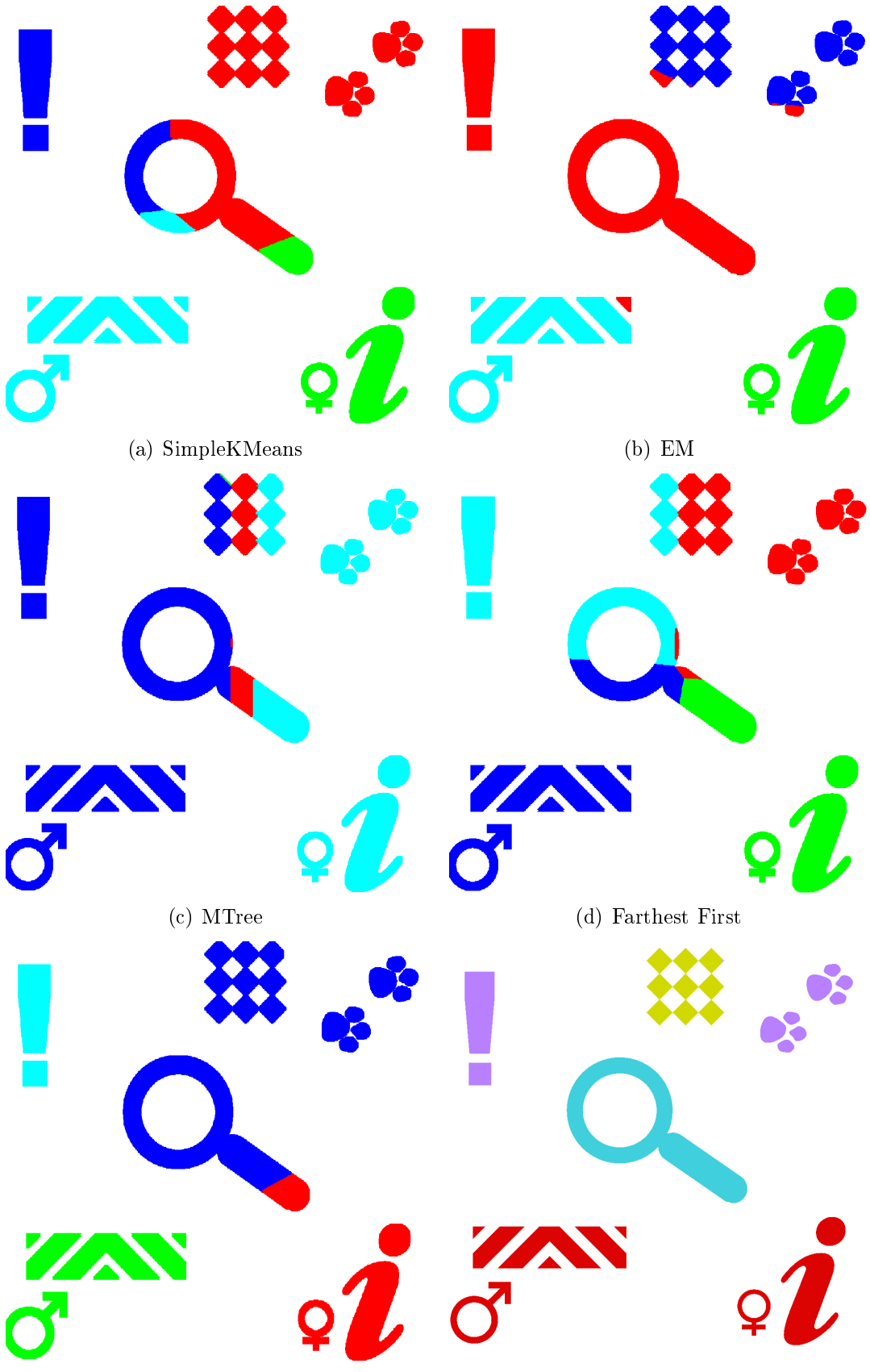
Table 4.6: Comparison of obtained results based on human volunteers.

Algorithms	Choice for k = 4 (%)	Error for k = 4 (%)	Choice for k = 8 (%)	Error for k = 8 (%)
SimpleKMeans	10.4	51.07	4.1	19.53
EM	6.3	37.46	16.3	20.22
MTree	4.2	59.05	4.1	59.02
Farthest First	12.5	50.49	2	27.07
Canopy	8.3	41.63	0	25.64
k-MS	**58.3**	0	**73.5**	0

values for k and various clustering algorithms that support the k variable in the Weka [36] framework, namely, (1) the k-Means implementation called SimpleKMeans [5], (2) Expectation Maximization (EM) [65], (3) MTree [21], (4) Farthest First [39] and (5) Canopy [60]. Figure 4.7 shows clusterization results for $k = 4$ and Figure 4.8 for $k = 8$. The different clusters are represented in different colors. The images are somewhat different because the one generated with k-MS was outputted by our algorithm and the remaining from the Weka visualization tool.

Figure 4.5 was presented to 73 volunteers, where they were instructed to answer the simple question: "If you were to separate this image in k groups, which image among the ones in Figures 4.7 and 4.8 would be the closest to your separation?". Two questions were asked for $k = 4$ and for $k = 8$, separately. Table 4.6 compares the average choice of these participants in percentage. The error column shows the average rate of incorrectly classified instances (in this case, pixels) in relation to the clusterization performed by our algorithm.

Given the results in Table 4.6, it is possible to infer that k-MS is the most "human-like" algorithm. For $k = 4$, the percentage of volunteers that voted for k-MS was 58.3%, which is significantly lower than for the $k = 8$ case (73.5%). This may be so due to the fact that, for $k = 4$, the exclamation mark on the left side was placed in the same cluster as the paws on the right side and this may have prevented volunteers from voting for k-MS in this occasion. In this group of experiments, dilations extrapolated the image boundaries on purpose to check how the volunteers would vote. Moving towards the right direction of the paws, the image boundary is extrapolated and the exclamation mark is reached faster than any other shape in the image. This is the exact reason for clustering together the exclamation mark and the paws in the $k = 4$ case. This could be avoided by conserving the image boundaries. In conclusion, the choice for k-MS was more frequent than for the remaining algorithms in both occasions regardless of the extrapolation.

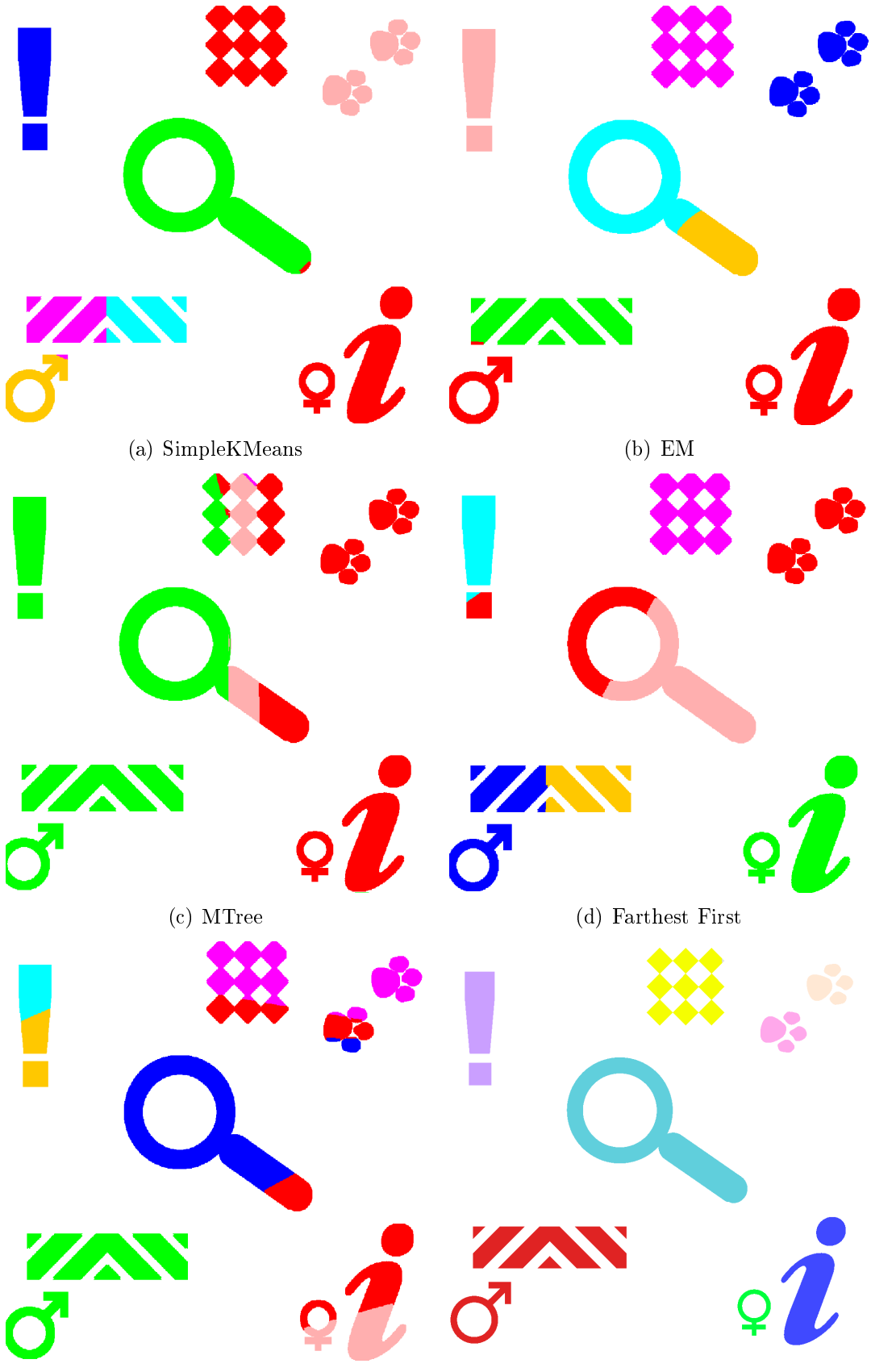


(a) SimpleKMeans (b) EM

(c) MTree (d) Farthest First

(e) Canopy (f) k-MS

Figure 4.7: Comparison of the visual results of k-MS to other clusterization algorithms for $k = 4$.



(a) SimpleKMeans (b) EM

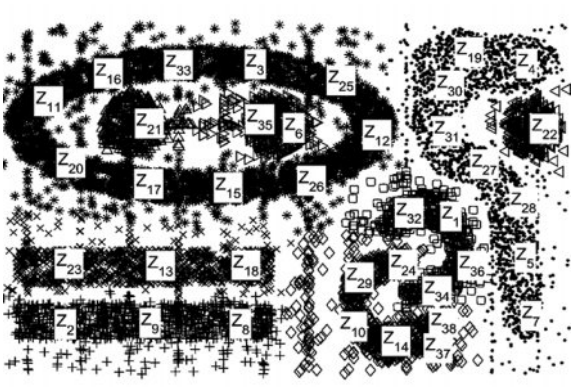
(c) MTree (d) Farthest First

(e) Canopy (f) k-MS

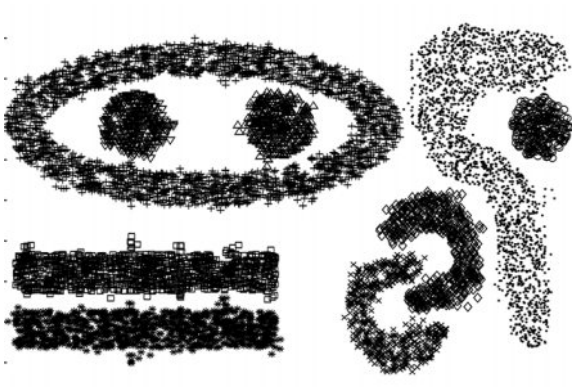
Figure 4.8: Comparison of the visual results of k-MS to other clusterization algorithms for $k = 8$.

4.2.2 Comparison with Similar Works

In this subsection, the clusterizations performed with k-MS, Chameleon, Mitosis, TRICLUST and M. Liu et al. algorithms on the dataset provided by Karypis et al. [44] are analysed and compared. In contrast to the previous Section 4.2.1, the clusterization algorithms are prepared to deal with density and shapes. Figure 4.9 shows the clusterizations obtained with each algorithm.



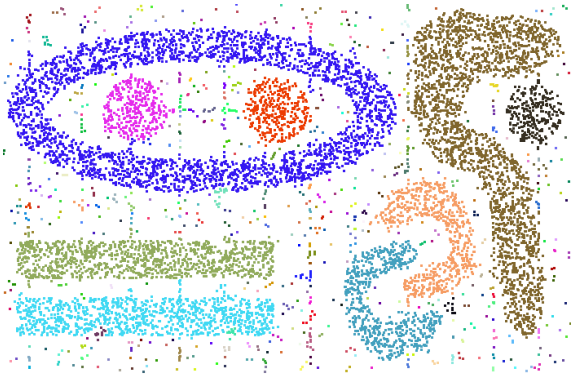
(a) M. Liu et al. [53]



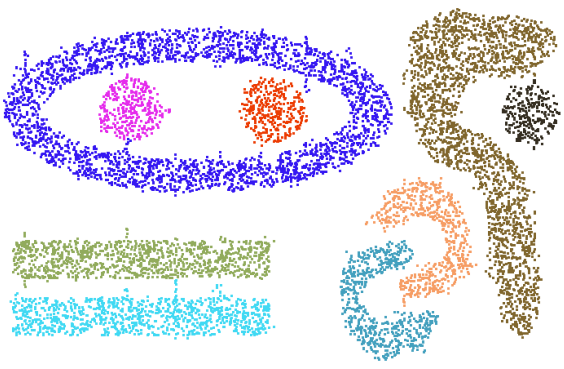
(b) TRICLUST [52]

(c) Mitosis [114]

(d) Chameleon [44]



(e) k-MS ($k = 450$)



(f) k-MS (Removed Noise)

Figure 4.9: Comparison of the visual results of k-MS and other clustering algorithms in finding genuine morphological clusters.

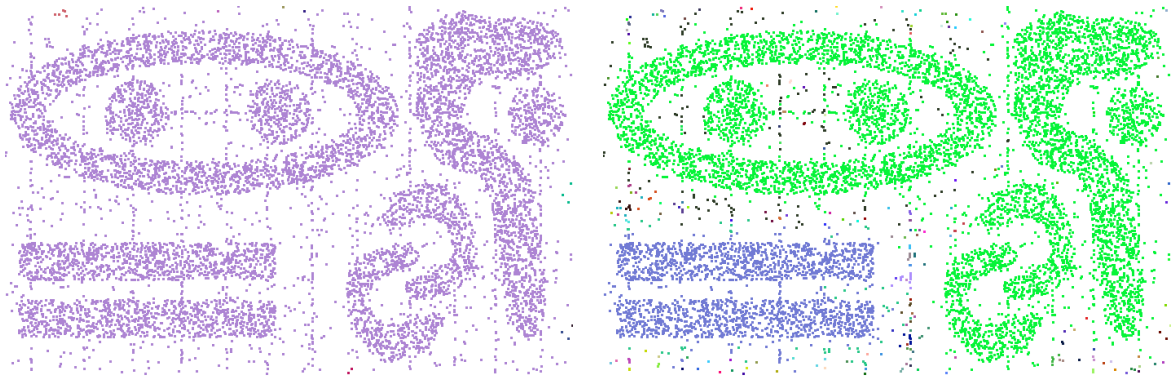
It can be seen in Figure 4.9 that the M. Liu et al. clusterization is difficult to visualize as the authors present it in their work. The symbols represent the different clusters and the letter Z in their image can be ignored. The authors of TRICLUST present the result without the noise of the initial dataset. Therefore, we cannot determine how well their algorithm would handle cluster noise.

The clusterizations of Mitosis and M. Liu et al. recognize the exact amount of genuine clusters (total of 9 clusters), such that all noise in the image is clusterized together with these genuine clusters. This outcome can be bad in some occasions such as when noise is expected to be removed or when it should not be aggregated in a genuine cluster. A practical example of this is image data that was collected by physical sensors.

On the other hand, the clusterization generated by Chameleon recognizes 9 genuine clusters as well as 3 other clusters that contain outliers. Chameleon and k-MS are the only two algorithms capable of segregating outliers in different clusters. k-MS goes even further, it indicates whether the clusters can be formed given a predefined k amount of clusters.

The result shown in Figure 4.9-(e) shows the clusters obtained with k-MS algorithm, where each cluster is depicted in a different color. In noisy datasets such as this one, the value of k should be large, otherwise k-MS would not be able to recognize the genuine clusters in the dataset. Furthermore, if the dataset is too sparse, it is also recommended to perform dilations on the dataset before applying k-MS, since the clusterization would be more accurate and converge faster.

When low values for k are considered, e.g., a maximum of 9 clusters, then the obtained result is shown in Figure 4.10-(a). In this case, the instances are too close to each other such that when dilated, several intersections occur, producing at the end of the processing the recognition of 9 clusters that are not the genuine ones. All genuine clusters were placed within a unique cluster in this case and the remaining were outliers (which cannot even be properly seen, as they are single points in this image).



(a) $k = 9$ (b) $k = 300$

Figure 4.10: Obtained clusters with lower values of k.

As k is increased to around 300, images like the one in Figure 4.10-(b) start to be generated. For $k = 300$, k-MS found 166 valid clusters, and the genuine clusters are now divided in two different clusters. Increasing k to 450 leads to the result shown in Figure 4.9-(e), where 450 valid clusters were found and the 9 genuine ones are separated correctly. In this case, k-MS took 6.33 seconds to generate these 450 valid clusters in the CPU.

The image shown in Figure 4.9-(f) is a version of (e) where all the noise is removed. This removal is fairly simple to be performed. Given a threshold τ, all clusters that contain τ or less pixels are erased. This is another particular and interesting property of k-MS. In the specific case of Figure 4.9-(e), τ was equal to 200 pixels.

It is also possible to change the structuring element to obtain different results. For instance, instead of using $B_1 = \{(0,0), (0,1), (0,-1), (1,0), (-1,0), (-1,-1), (1,-1), (-1,1), (1,1)\}$ which is shown in Figure 3.9, $B_2 = \{(0,0), (0,1), (0,-1), (1,0), (-1,0), (-10,-10), (10,-10), (-10,10), (10,10)\}$ could be used instead, where the diagonal pixels are more distant from the origin than others. For $k = 450$ and B_2 the result shown in Figure 4.11 is obtained. In this case, 372 valid clusters were found and the running time was at least 400 times faster than with B_1. Several combinations of clusters can be generated by tweaking the two parameters of k-MS.

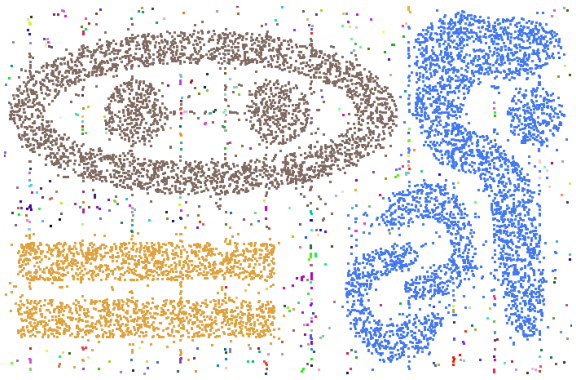


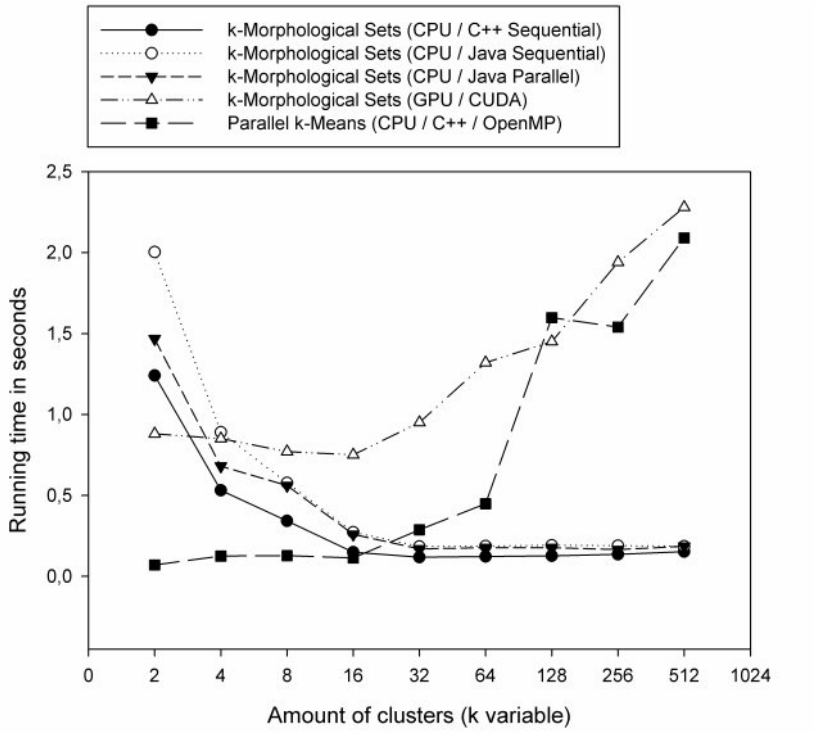
Figure 4.11: Result for $k = 450$ and B_2.

At last, all the addressed algorithms were able to recognize the genuine clusters in this dataset. Due to this fact, we did not ask volunteers to choose a best segmentation, as volunteers would be overly influenced by the different types of images that each author adopted.

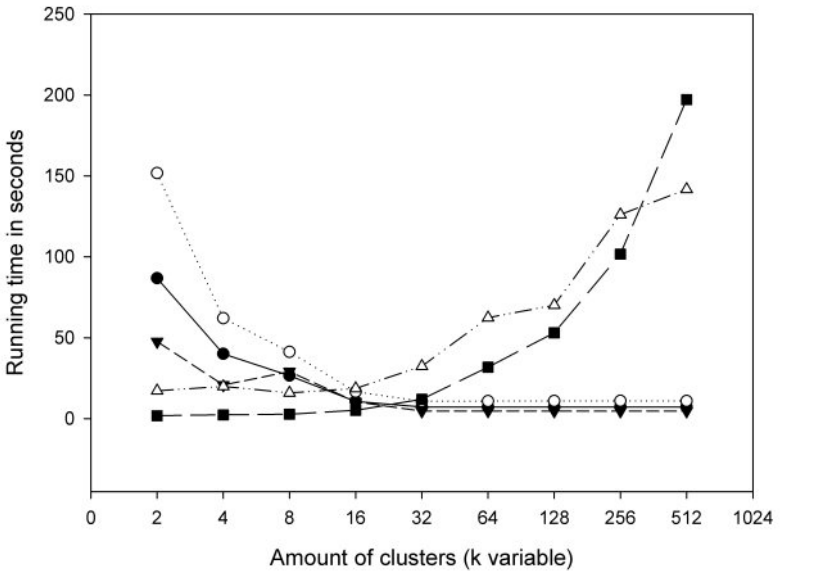
4.2.3 Run Time Analysis

An extensive time analysis experiment is performed in this section. In order to compare and measure the time performance on datasets with a large amount of instances, we considered the GPU-oriented k-Means algorithm presented in [31], which is based on [50] and [57]. However, this implementation could not handle the size of the instances we evaluated, which is a positive aspect of k-MS. Therefore, our comparisons on this section address just the implementations of [50], which is a parallel k-Means for the CPU using OpenMP [1]. We tested the sequential k-Means provided by them as well but it was always slower than the parallel one, and therefore it was disregarded.

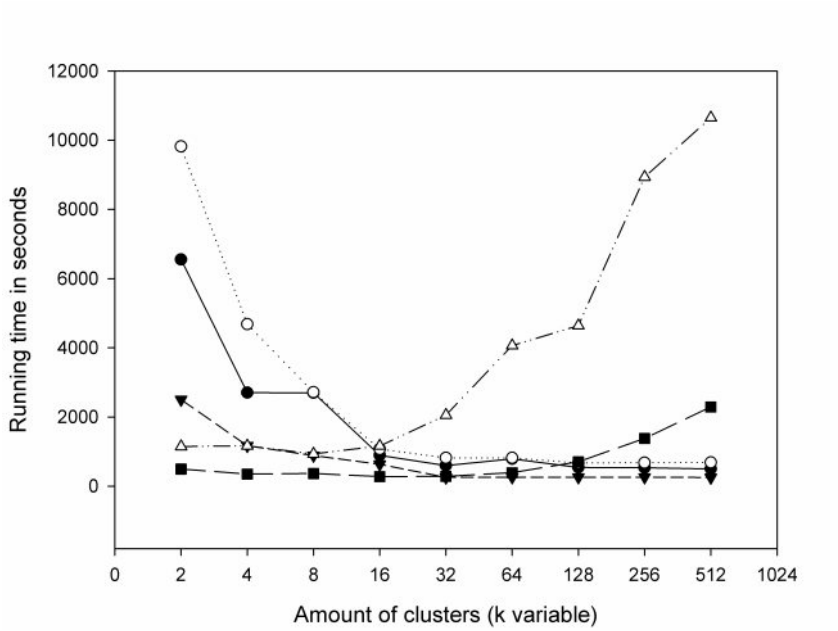
The threshold variable of the Liao algorithm [50] was set to 10^{-12}, which essentially means that the algorithm stops when less than $10^{-10}\%$ of the instances change membership, i.e., change clusters. Since this is a low number, we are essentially setting the algorithm to converge when no instance changes membership. The comparisons are shown in Figure 4.11, for instances of 512×512, 1024×1024 and 4096×4096, respectively. The presented results, however, would vary depending on the computer specification. A Quad-core Intel i7-4720HQ, with 8 threads in OpenMP and in Java, and a Nvidia GTX 960M were used in these experiments.



(a) 512×512 image, containing a total of 36,529 data points or instances.



(b) 2048×2048 image, containing a total of 584,235 data points or instances.



(c) 8192 × 8192 image, containing a total of 9,352,084 data points or instances.

Figure 4.11: Time comparison between k-MS and a parallel k-Means considering a significant amount of instances.

It is possible to see that k-Means starts faster for low values of k but is overcome in every occasion after a particular size of k. As the size of k increases, so does the running time of clusterization algorithms in general. However, due to the early break in k-MS, the running times obtained with k-MS may even decrease, regarding the CPU, as k increases. This happens because the clusterization converges faster with higher values of k (less morphological reconstructions), and the early breaks reduce the time of the clusters verification. Furthermore, the GPU implementation is only viable if the amount of instances is very large and k is small. Small values of k reduce the amount of atomic operations that have to be performed, which reduces the processing time of the algorithm.

Table 4.7 shows the running times obtained with k-MS for smaller amounts of instances. This comparison is necessary because no reported time performances of these algorithms considering a significantly high number of instances is present in the literature. The first column of the table indicates the values of k, and the first line indicates the amount of instances. In this case, just the CPU C/C++ sequential implementation was considered for a fair comparison.

Table 4.7: Run time performance (s) for the k-MS algorithm as a function of the number of instances and maximum cluster number k.

k	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000	11000	12000
2	0.147	0.066	0.125	**0.074**	**0.079**	**1.517**	**0.902**	**0.752**	**0.747**	**0.765**	**0.739**	**0.740**
4	0.156	0.069	0.128	0.074	0.856	1.537	0.975	0.812	0.804	0.797	0.800	0.797
8	0.153	0.072	0.136	0.086	0.891	1.659	1.087	0.859	0.842	0.865	0.838	0.861
16	0.129	0.081	0.110	0.089	0.105	1.856	1.179	0.960	0.957	0.962	0.969	0.965
32	0.113	0.108	0.140	0.126	0.135	2.329	1.557	1.331	1.458	1.323	1.323	1.251
64	0.133	0.132	0.117	0.156	0.119	3.104	2.118	1.818	1.853	1.867	1.708	1.673
128	0.090	0.143	**0.084**	0.118	0.094	3.735	2.635	2.234	2.469	2.347	2.146	2.109
256	**0.052**	0.079	0.125	0.090	0.094	2.710	3.478	1.987	3.359	2.950	1.397	1.538
512	0.056	**0.061**	0.097	0.096	0.118	2.668	1.504	2.048	1.680	1.127	1.045	1.606

The times in bold represent the best times in the column, while the ones highlighted in red represent the worst.

D. Liu et al. [52] provide a plot in their work that contains the running times with different numbers of instances. Yousri et al. [114] provide four plots with the same intent, where each corresponds to a different dataset. Just these two works, among [44, 52, 53, 114, 22, 47, 118, 116, 115], provide a time analysis while varying the amount of instances. We approximated the times shown in these plots, and compared them to the worst times that our algorithm achieved (the ones highlighted in red in Table 4.7). The comparison can be seen in Figure 4.12.

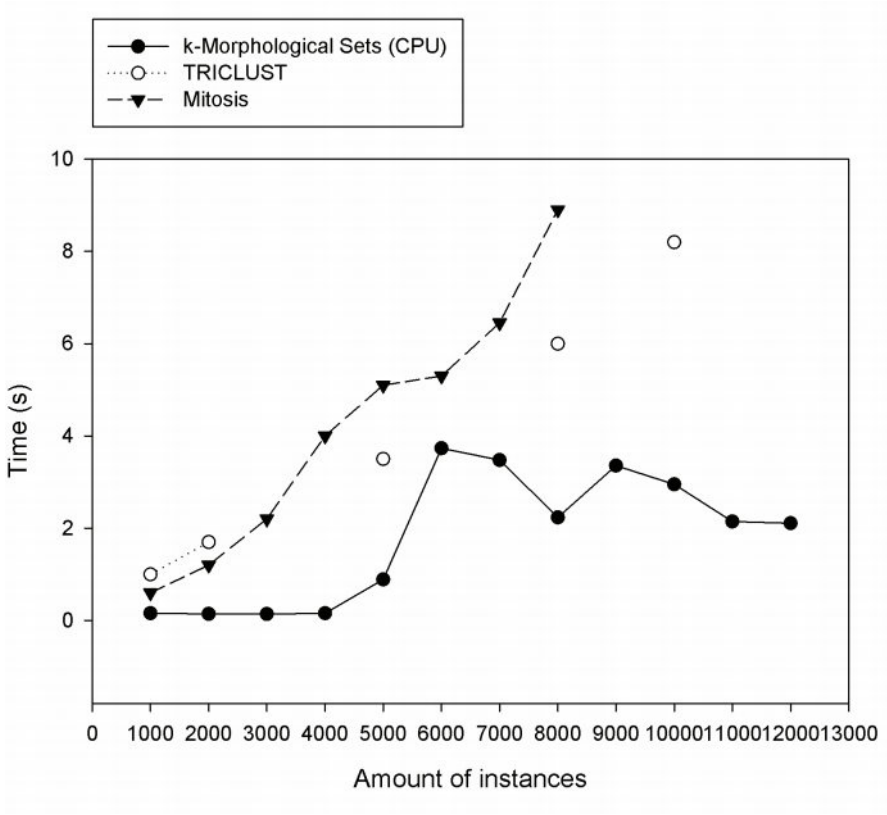


Figure 4.12: Run times (s) comparison for varying numbers of instances.

This comparison is not fair due to several reasons, which includes the fact that the used datasets are different. In our case, images that had the 512×512 proportion were used, where n instances were generated at random places. Yousri et al. [114] provide four plots for comparing running times, the one with the lowest values was chosen, while in our case, the highest values in Table 4.7 were selected, such that Yousri et al. algorithm (Mitosis) as well as TRICLUST are favoured. Still, k-MS was faster than both Mitosis and TRICLUST in every occasion.

4.3 Summary

This chapter presented results and experimental analyses regarding classification and clusterization when using morphological operators. When it comes to classification, proposed algorithms were able to provide good accuracy rates even when considering p-dimensional datasets (outperforming competitive classifiers), which indicates, as initial evidence, that the use of morphology in classification can be fertile in several aspects. At total, we provide four different implementations of two algorithmic proposals, each one considering slightly different premises.

Regarding the clusterization part, we were able to provide a very unique clustering algorithm that is able to recognize density and cluster shapes. This algorithm ended up being faster than all state-of-the-art clustering algorithms that are sensible to cluster density and shapes. In this comparison, we considered all works that used the dataset shown in Figure 4.9, that we were able to find, for a proper visual comparison. It is clear that, regarding both approaches, mathematical morphology is at least feasible in this matter.

Chapter 5

Conclusion

This thesis focused on introducing mathematical morphology-based techniques to machine learning, which has never been properly addressed in the literature. To the best of our knowledge, no classifier that uses mathematical morphology exists, with the exception of image classification that uses binary operations. Clustering methods that use mathematical morphology are present in the literature, though in a very limited number. We cover all of them in the literature review chapter. These methods are not entirely based on mathematical morphology and do not use grey level operations. These are mostly based on BMCAs.

Two classification algorithms (4 variants at total) and one clusterization algorithm were proposed and described in this thesis. In the classification case, the proposed algorithms (MDC and MkNN) tied or outperformed other 14 well established classifiers in 5 out of 8 p-dimensional UCI datasets. In all of these cases, the proposed classifiers obtained a higher accuracy than the accuracy averaged across all classifiers.

The clusterization proposal is a whole different case. k-MS outperformed all state-of-the-art clustering algorithms that are sensitive to cluster shape and density in terms of processing times. Besides, it is the only algorithm that has an intrinsic sense of maximal clusters that can be created. Given a predefined k variable, the algorithm outputs either k or less clusters, depending on the data configuration. When density is regarded, it is not possible to separate high density clusters in some occasions. Let us suppose connected data points forming a shape of a circle. In this case, if $k > 1$, the algorithm would still segregate the data in a unique cluster. In addition, it is also the only algorithm in the literature that provides a noise removal paradigm without requiring extra processing costs.

An extra major contribution consists of a metric that was coined on the process of iterating image neighbourhoods using GPUs. GPUs rely on low level instruction environments and therefore complex data structures are not practical. This proposed distance is closer to Euclidean in terms of neighbourhood iteration, when compared to Chebyshev and Manhattan, while being faster to compute. In fact, when plotted in $\mathbb{R}^2$, the proposed metric displays an octagon, which is closer to the circle displayed by the Euclidean metric. As the proposal consists of a sum of two metrics, we prove that it a metric as well.

It is substantial to argue that the use of morphology in machine learning is feasible and may be highly fertile in the future, when algorithms of this kind are further improved.

5.1 Discussion

Classification experiments presented herein focus on proposing and evaluating 2-dimensional models and further aggregating them to form p-dimensional classification models, where p stands for the number of features or dimensions of the dataset. Although our theoretical framework and proposed algorithms readily work for p-dimensional datasets, we proved that it is faster, in terms of the number of required operations in the worst-case scenario, to generate 2-dimensional classification models and to combine them, rather than working in p-dimensional spaces and creating p-dimensional classification models. These 2-dimensional models are combined using a voting scheme, originating an ensemble classifier.

The complexity for checking whether an instance belongs to a certain class using matrices is $O\binom{p}{2}$ in the worst case, using the combination of 2D models. This is a very fast instance classification that is not dependant on the number of instances in the dataset. The classification time is potentially faster than decision-tree based classification, which are already fairly fast in comparison to other classification paradigms.

Drawbacks include the fact that training morphological classifiers may be slow, particularly when dilations are performed in p-dimensions directly. However, due to the nature of the operations performed, such classifiers are amenable to parallelization. Moreover, in a fashion similar to the training of deep learning networks, slow training times may not be considered a major disadvantage, as the actual classification process is almost instantaneous and classification models demonstrate reasonable accuracy rates.

In what concerns to the clusterization proposal, k-MS is substantially different from all state-of-the-art clusterization algorithms due to: (1) having a sense of maximal clusters

that can be created, (2) being able to be efficiently implemented in parallel, (3) permitting the use of structuring elements that can alter the way that clusters propagate and are created, (4) providing a fast and simple means for removing noise with no extra processing costs and, (5) due to its simplicity. In addition, k-MS is also faster than all the remaining evaluated algorithms, including sequential and parallel k-Means, TRICLUST [52] and Mitosis [114].

Although the k-MS proposal works with p-dimensional datasets, we optimized the implementation used in this work for images. This implementation can be further extended to work with p-dimensional datasets and can be applied in several fields such as in data mining, machine learning and other areas of computer vision. As a final remark, all the source code and datasets used in this work are available in [80, 81].

5.2 Future Work

Other possibilities to explore in the morphological classifiers regard is introducing more rules for the expansion fashion of MDC. The current MDC proposal is fairly naive, and that naiveness certainly produces more classification errors than necessary. Currently, the fashion of growth is limited and does not encompass complex directions. Furthermore, different structuring elements, growth rules and distances (for the neighbourhood iteration) could be used in each 2D model, which has not been evaluated.

Dynamic structuring elements are also a potential improvement. That is, the dynamic word indicates structuring elements that are modified during the classification model construction. This can be used to generate or mimic a fractal growth, for instance. These structuring elements can be modified using predefined heuristics.

We attempted to grow models in a fractal fashion. This produced acceptable results, but this fractal approach was too simple and it did not outperform most classifiers, being worse than the current approach described in this thesis. However, the idea of exploring fractal information in datasets has not yet been explored. The proposed morphological framework allows for this possibility, it just requires further enhancements to properly deal with it.

Another clear avenue is to explore algorithms that are dilated in p-dimensions using p-dimensional morphological operations such as p-dimensional dilations, conditional thickening [61] and others, in an efficient approach. p-dimensional dilation experiments have been carried, but they are fairly difficult to be implemented efficiently (in terms

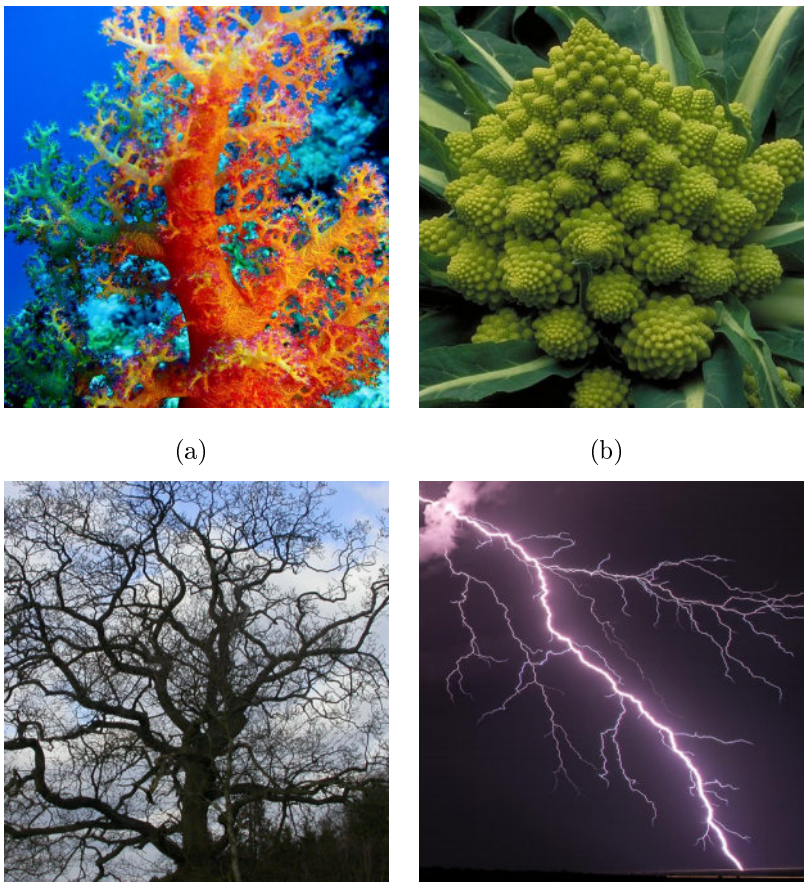
of memory storage and processing times). Nevertheless, p-dimensional approaches are still feasible. Future research may focus on reducing these burdens by using approximations (such as prototypes or cluster-cores) and other heuristics, enabling p-dimensional morphological approaches to be performed in real time for a broad extent of datasets.

In addition, data compression techniques can be applied to reduce the burden of memory requirements. We aim to further explore the application of such techniques and to propose efficient ways of storing classification models regarding large datasets in future works. It is necessary to highlight, however, that the combination approach using 2D models works well even with large datasets, as memory and processing burdens are mitigated.

When it comes to clusterization, future work involve porting, and evaluating to a broader extent, the clusterization algorithm proposed in this work. Fractal based growths can also be explored, which has not been attempted yet. Similarly to the classification case, dynamic structuring elements and more complex mathematical morphology techniques such as conditional thickening [61], skeletonization [75], and others, could be applied.

5.2.1 Fractal Approach

In this last section, we provide an overview of what could be done using fractals for constructing classification models. Attractive and inspirational images can be constructed using principles from chemistry and physics. One of such examples is the diffusion limited aggregation (DLA), which describes the diffusion and aggregation of zinc ions in an electrolytic solution onto electrodes. Diffusion refers to the fact that particles forming structures wander around randomly before attaching themselves (aggregation) to the structure. Limited refers to the fact that particles are expected to be in low concentrations such that they do not come in contact with each other and the structure grows one particle at a time [9]. One of the first studies on this matter was introduced by Witten et al. [112]. DLA represents fractal behaviour, which is very frequent in nature. Other examples can be found in coral growth (Figure 5.1-a), vegetables (Figure 5.1-b), trees (Figure 5.1-c), lightning (Figure 5.1-d), etc.

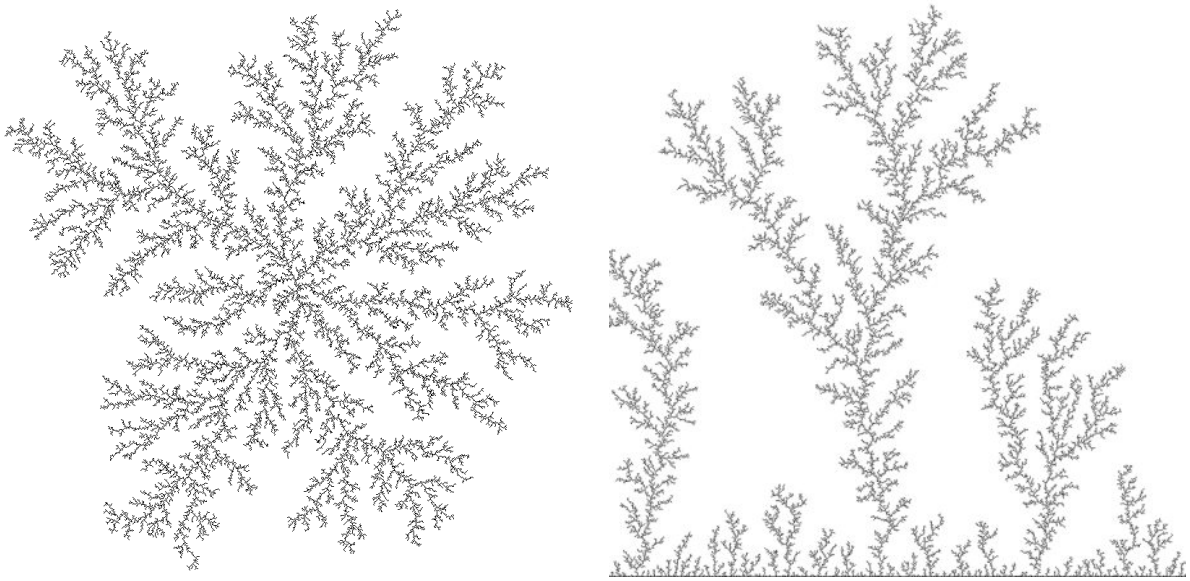


(a) (b)

(c) (d)

Figure 5.1: Fractal patterns in nature.

Similar fractal processes can be simulated in a number of different ways in computer environments. As a practical example, an image with a single black pixel in its center is created. Later, new points are introduced at the borders, which randomly walk through the image until they are close enough to stick to a pre-existent pixel. Figure 5.2-(a) illustrates this process. Aternatively, a line can also be used as an attractor, instead of a single point. New pixels are introduced from the top of the image while the boundaries of the images are disregarded, i.e., if points move off the left boundary they appear on the right and vice-versa. Figure 5.2-(b) illustrates this second example.

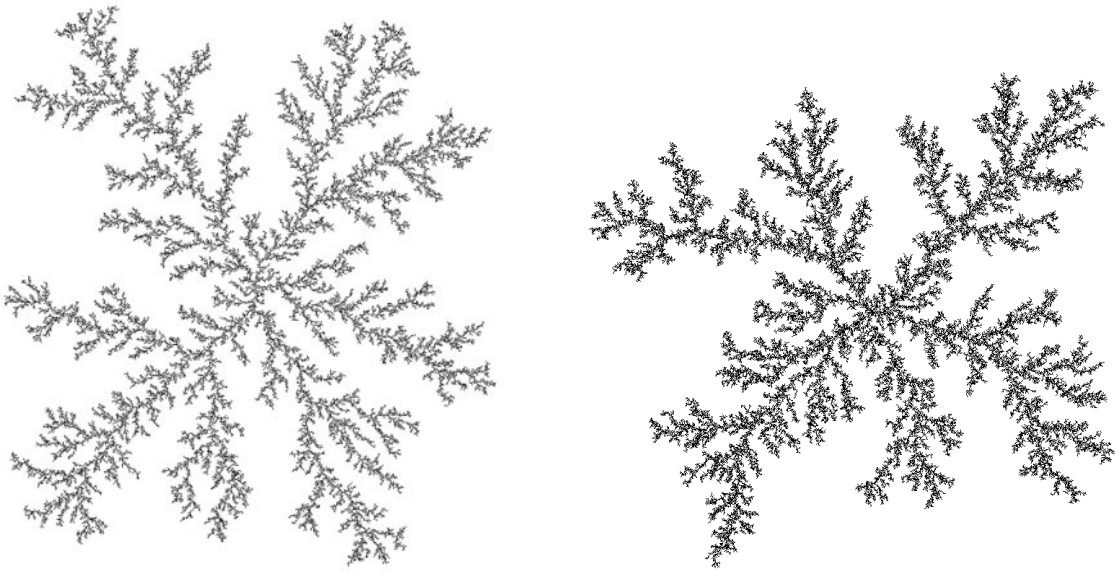


(a) Point attractor at the center.

(b) Line attractor at the bottom.

Figure 5.2: Creation of fractal structures using DLA [9].

It is also possible to vary the approach aiming to change the thickness of the produced fractals. A stickiness probability can be given to new points, which indicate how likely is their attachment to the contacted pixel. Figure 5.3 illustrates the idea. Figure 5.3-(a) represents a higher probability of sticking to previously existent pixels than the one shown in Figure 5.3-(b).

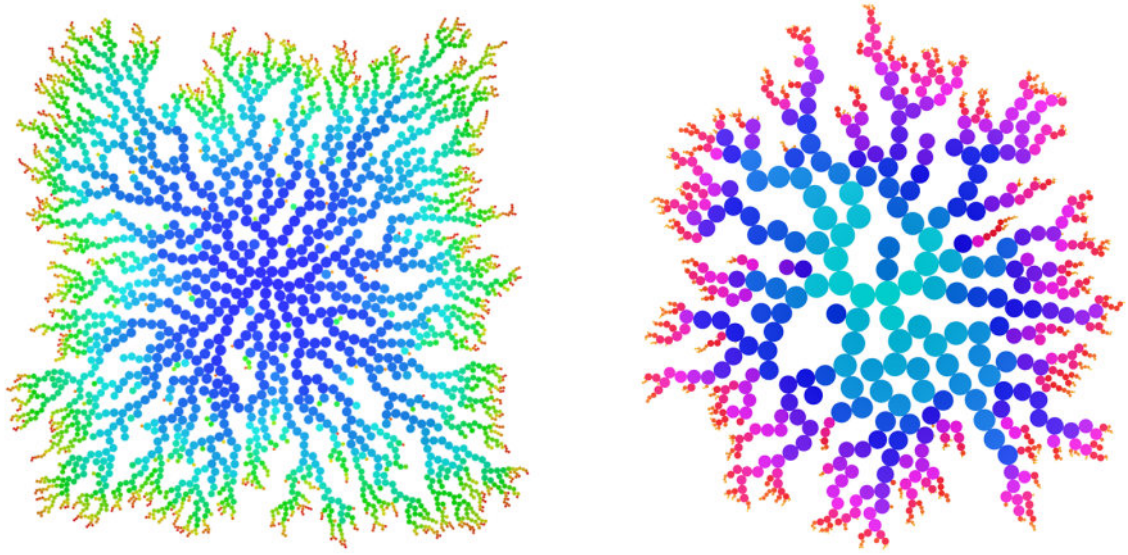


(a)

(b)

Figure 5.3: Thicker structures using a stickiness probability [9].

Spheres of different sizes can also be used instead of single points. If we colour them by the order in which they stick to previous elements, we would have results such as the ones shown in Figure 5.4. Larger spheres can be used in more dense areas, or as a means of approximation, while small spheres can be used in less dense subspaces.



(a) Fractal formed with spheres of different sizes. (b) A second potential outcome using spheres of different sizes.

Figure 5.4: Structures drawn using spheres of different sizes [9].

It may be clear by now how this described process is related to the classification approach. Methodologies similar to these could be used to formulate a classification model. Training points can represent these initial attractor points (one label should be processed at a time). As time advances, new elements that randomly walk through the classification space, stick to this growing structure. The misclassification error should be computed at each change of the model and, if necessary, the iteration is halted. The result of this process is used to predict the label of unlabelled instances. That is, if an unlabelled instance belongs to an arbitrary fractal structure, then it most probably belongs to the same class of points that form the initial structure.

Although very inspiring, this approach bears significant difficulties in terms of efficiency. Furthermore, it is very difficult to implement an algorithm of such kind. In p-dimensional spaces, how would we define the path of new points? Heuristics could be used, but it is still extremely costly. Either combinations of 2D images should be used, or approximations of these paths must be developed to reduce the time burden. Future works will also focus on this described approach.

References

[1] Openmp, 2017. Available at `http://www.openmp.org/`.

[2] Abbadi, E.; Khdhair, N.; Saadi, A.; Hamood, E. Blood vessels extraction using mathematical morphology. *Journal of Computer Science 10* (2013), 1389–1395.

[3] Ali, A. R.; Couceiro, M. S.; Hassanien, A. E.; Hemanth, D. J. Fuzzy c-means based on minkowski distance for liver ct image segmentation. *Intelligent Decision Technologies 10*, 4 (2016), 393–406.

[4] Altman, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician 46* (1992), 175–185.

[5] Arthur, D.; Vassilvitskii, S. k-means++: the advantages of carefull seeding. *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (2007), 1027–1035.

[6] Bai, X.; Zhou, F. Analysis of new top-hat transformation and the application for infrared dim small target detection. *Pattern Recognition 43* (2010), 2145–2156.

[7] Bicego, M.; Cristani, M.; Fusiello, A.; Murino, V. Watershed-based unsupervised clustering. *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition* (2003), 83–94.

[8] Bien, J.; Tibshirani, R. Prototype selection for interpretable classification. *The Annals of Applied Statistics 5* (2011), 2403–2424.

[9] Bourke, P. Dla - diffusion limited aggregation, 2014. Available at `http://paulbourke.net/fractals/dla/`.

[10] Breiman, L. Random forests. *Machine Learning 45* (2001), 5–32.

[11] Broomhead, D. S.; Lowe, D. Multivariable functional interpolation and adaptive networks. *Complex Systems 2* (1988), 321–355.

[12] Buhmann, M. D. *Radial Basis Functions: Theory and Implementations*. Cambridge University, 2003.

[13] Castillo, E.; Gutierrez, J.; Hadi, A. *Learning Bayesian Networks*. 1997.

[14] Cha, S. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences* (2007), 300–307.

[15] Chen, C. S.; Yeh, C. W. An efficient dilation-based clustering algorithm for automatic optical inspection. *The 11th International Conference on Information Sciences, Signal Processing and their Applications* (2012).

[16] Chen, T.; Wu, Q. H.; Torkaman, R. R.; Hughes, J. A pseudo top-hat mathematical morphological approach to edge detection in dark regions. *Pattern Recognition 35* (2002), 199–210.

[17] Conci, A.; Azevedo, E.; Leta, F. R. *Computacao Grafica: Teoria e Pratica.* Elsevier, 2003.

[18] Conci, A.; Kubrusly, C. S. Distance between sets: A survey. *Advances in Mathematical Sciences and Applications 17* (2017), 1343–4373.

[19] Conci, A.; Saade, D. C. M.; GalvÃČo, S. S. L.; Sequeiros, G. O.; MacHenry, T. A new measure for comparing biomedical regions of interest in segmentation of digital images. *Discrete Applied Mathematics 197* (2015), 103–113.

[20] Cortes, C.; Vapnik, V. Support-vector networks. *Machine Learning 20* (1995), 273.

[21] Cristian, M.; Dumitru, M.; Burdescu, D. Using m tree data structure as unsupervised classification method. *Informatica (Slovenia) 36* (2012), 153–160.

[22] Daneshgar, A.; Javadi, R.; Razavi, S. B. S. Clustering and outlier detection using isoperimetric number of trees. *Pattern Recognition 46* (2013), 3371–3382.

[23] Dominguez-Lopez, J.; Damper, R.; Crowder, R.; Harris, C. Adaptive neurofuzzy control of a robotic gripper with on-line machine learning. *Robotics and Autonomous Systems 48* (2004), 93–110.

[24] Dougherty, E. R. *An Introduction to Morphological Image Processing.* Society of Photo Optical, 1992.

[25] Dufour, A.; Tankyevych, O.; Naegel, B.; Talbot, H.; Ronse, C.; Baruthio, J.; DoklÃądal, P.; Passat, N. Filtering and segmentation of 3d angiographic data: Advances based on mathematical morphology. *Medical Image Analysis 17* (2013), 147–164.

[26] Facon, J. A morfologia matemática e suas aplicacões em processamento de imagens. *VII Workshop de Visao Computacional* (2011).

[27] Fayyad, U.; Piatetsky-Shapiro, G.; Smyth, P.; Uthurusamy, R. Trends in data mining and knowledge discovery. *Advances in Knowledge Discovery and Data Mining* (1996).

[28] Fisher, D. L. Data, documentation, and decision tables. *Communications of the ACM 9* (1966), 26–36.

[29] Gan, G.; Ma, C.; Wu, J. *Data clustering: theory, algorithms, and applications*, vol. 20. Siam, Series on Estatistics and Applied Mathematics, 2007.

[30] Geraud, T.; Strub, P. Y.; Darbon, J. Color image segmentation based on automatic morphological clustering. *Proceedings of Image Processing* (2001).

[31] Giuroiu, S. Cuda k-means clustering. http://serban.org/software/kmeans/.

[32] Goldsmith, J. Unsupervised learning of the morphology of a natural language. *Computational Linguistics 27* (2001), 153–198.

[33] Gonzalez, R. C.; Woods, R. E.; Eddins, S. L. Morphological reconstruction: From digital image processing using matlab. Available at `http://mathworks.com/tagteam/64199_91822v00_eddins_final.pdf`.

[34] Goswami, S.; Bhaiya, L. K. P. Brain tumour detection using unsupervised learning based neural network. *Communication Systems and Network Technologies* (2013), 573–577.

[35] Greche, L.; Jazoui, M.; Es-Sbai, N.; Majda, A.; Zarghili, A. Comparison between euclidean and manhattan distance measure for facial expressions classification. *Wireless Technologies, Embedded and Intelligent Systems (WITS)* (2017).

[36] Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I. H. The weka data mining software: An update. *SIGKDD Explorations 11* (2009).

[37] Halmos, P. R. *Naiva Set Theory.* Springer, 1998.

[38] Haykin, S. *Neural Networks: A Comprehensive Foundation.* Prentice Hall, 1998.

[39] Hochbaum, S. A best possible heuristic for the k-center problem. *Mathematics of Operations Research 10* (1985), 180–184.

[40] Huang, L.; Kim, H.; Furst, J.; Raicu, D. A run-length encoding approach for path analysis of c. elegans search behavior. *Comput Math Methods Med* (2016).

[41] Hulten, G.; Spencer, L.; Domingos, P. Mining time-changing data streams. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* (2001), 97–106.

[42] Jonker, P. P.; Svensson, S. The generation of n dimensional shape primitives. *Lecture Notes in Computer Science 2886* (2003), 420–433.

[43] Kamimura, R.; Uchida, O. Greedy network-growing by minkowski distance functions. *Proceedings of the 2004 IEEE International Joint Conference on Neural Networks* (2004).

[44] Karypis, G.; Han, E. H.; Kumar, V. Chameleon: Hierarchical clustering using dynamic modeling. *Computer 32* (1999), 68–75.

[45] Koenig, S.; Simmons, R. G. Unsupervised learning of probabilistic models for robot navigation. *Proceedings of the IEEE International Conference on Robotics and Automation 3* (1993), 1050–4729.

[46] Kohavi, R. The power of decision tables. *Proceedings of the European Conference on Machine Learning* (1995), 174–189.

[47] Kumar, K. M.; Reddy, A. R. M. A fast dbscan clustering algorithm by accelerating neighbor searching using groups method. *Pattern Recognition 58* (2016), 39–48.

[48] Lance, G. N.; Williams, W. T. Computer programs for hierarchical polythetic classification. *The Computer Journal 9*, 1 (1966), 60–64.

[49] Lee, P. M. *Bayesian Statistics*. Wiley, 2012.

[50] Liao, W. Parallel k-means data clustering. Available at `http://users.eecs.northwestern.edu/~wkliao/Kmeans/index.html`.

[51] Likas, A.; Vlassis, N.; Verbeek, J. J. The global k-means clustering algorithm. *Pattern Recognition 36* (2003), 451–461.

[52] Liu, D.; Nosovskiy, G. V.; Sourina, O. Effective clustering and boundary detection algorithm based on delaunay triangulation. *Pattern Recognition Letters 29* (2008), 1261–1273.

[53] Liu, M.; Jiang, X.; Kot, A. C. A multi-prototype clustering algorithm. *Pattern Recognition 42*, 5 (2009), 689–698.

[54] Liu, M.; Liu, Y.; Hu, H.; Nie, L. Genetic algorithm and mathematical morphology based binarization method for strip steel defect image with non-uniform illumination. *Journal of Visual Communication and Image Representation 37* (2016), 70–77.

[55] Lucena, M.; Martinez-Carrillo, A. L.; Fuertes, J. M.; Carrascosa, F.; Ruiz, A. Decision support system for classifying archaeological pottery profiles based on mathematical morphology. *Multimed Tools Appl 75* (2016), 3677.

[56] Luo, H.; Kong, F.; Zhang, K.; He, L. A clustering algorithm based on mathematical morphology. *Proceedings of the 6th World Congress on Intelligent Control and Automation* (2006).

[57] MacQueen, J. Some methods for classification and analysis of multivariate observations. *Berkeley Symposium on Mathematical Statistics and Probability 1* (1967), 281–297.

[58] Matheron, G.; Serra, J. The birth of mathematical morphology.

[59] Matzkevich, I.; Abramson, B. The topological fusion of bayes nets. *UAI'92 Proceedings of the Eighth international conference on Uncertainty in artificial intelligence* (1992), 191–198.

[60] McCallum, A.; Nigam, K.; Ungar, L. Efficient clustering of high dimensional data sets with application to reference matching. *Proceedings of the sixth ACM SIGKDD internation conference on knowledge discovery and data mining ACM-SIAM symposium on Discrete algorithms* (2000), 169–178.

[61] Meyer, F. Automatic screening of cytological specimens. *Computer Vision, Graphics, and Image Processing* (1986), 356–369.

[62] Meyer, F.; Maragos, P. Multiscale morphological segmentations based on watershed, flooding, and eikonal pde. *International Conference on Scale-Space Theories in Computer Vision* (1999), 351–362.

[63] Michalski, R. S. Knowledge acquisition through conceptual clustering: A theoretical framework and an algorithm for partitioning data into conjunctive concepts. *Policy Analysis and Information Systems 4* (1980), 219–244.

[64] Mlynarczuk, M. Description and classification of rock surfaces by means of laser profilometry and mathematical morphology. *International Journal of Rock Mechanics and Mining Sciences 47* (2010), 138–149.

[65] Moon, T. K. The expectation-maximization algorithm. *IEEE Signal Processing Magazine 13* (1996), 47–60.

[66] Morik, K. Applications of machine learning. *Current Developments in Knowledge Acquisition 599* (2005), 9–13.

[67] Opitz, D.; Maclin, R. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research 11* (1999), 169–198.

[68] Ortiz, F.; Torres, F. Vectorial morphological reconstruction for brightness elimination in colour images. *Real-Time Imaging 10* (2004), 379–387.

[69] Pedrino, E. C.; Nicoletti, M. C.; Saito, J. H.; Cura, L. M. V.; Roda, V. O. A binary morphology-based clustering algorithm directed by genetic algorithm. *IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (2013).

[70] Pentaho. Data mining community documentation. http://wiki.pentaho.com/display/DATAMINING, 2016.

[71] Pina, P.; Barata, T. Classification by mathematical morphology. *Proceedings of the 2003 IEEE International Geoscience and Remote Sensing Symposium* (2003).

[72] Pitsoulis, L.; Resende, M. *Greedy randomized adaptive search procedures.* Handbook of Applied Optimization, 2003.

[73] Platt, J. C. Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Learning* (1998), 185–208.

[74] Postaire, J. G.; Zhang, R. D.; Botte, L. Cluster analysis by binary morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15* (1993), 170–180.

[75] Pudney, C. Distance-ordered homotopic thinning: A skeletonization algorithm for 3d digital images. *Computer Vision and Image Understanding* (1998), 404–413.

[76] Quinlan, J. R. *C4.5: programs for machine learning.* Morgan Kaufmann Publishers Inc., 1993.

[77] Quinlan, J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 2014.

[78] Rish, I. An empirical study of the naive bayes classifier. Tech. rep., 2001.

[79] Rodrigues, E.; Morais, F.; Morais, N.; Conci, L.; Neto, L.; Conci, A. A novel approach for the automated segmentation and volume quantification of cardiac fats on computed tomography. *Computer Methods and Programs in Biomedicine* (2015).

[80] Rodrigues, E. O. k-morphological sets: Source and datasets. Available at `https://github.com/Oyatsumi/kMorphologicalSets`.

[81] Rodrigues, E. O. Morphological classifiers source code. https://github.com/Oyatsumi/Morphological-Classifiers/tree/master, 2016.

[82] Rodrigues, E. O.; Clua, E. A real time lighting technique for procedurally generated 2d isometric game terrains. *Entertainment Computing - ICEC 2015 9353* (2015), 32–44.

[83] Rodrigues, E. O.; Conci, A.; Borchartt, T. B.; Paiva, A. C. Comparing results of thermographic images based diagnosis for breast diseases. *Proceedings of IWSSIP* (2014), 39–42.

[84] Rodrigues, E. O.; Conci, A.; Morais, F. F. C.; Perez, M. G. Towards the automated segmentation of epicardial and mediastinal fats: A multi-manufacturer approach using intersubject registration and random forest. *IEEE International Conference on Industrial Technology (ICIT)* (2015), 1779–1785.

[85] Rodrigues, E. O.; Morais, F. F. C.; Conci, A. On the automated segmentation of epicardial and mediastinal cardiac adipose tissues using classification algorithms. *MEDINFO 2015: EHealth-enabled Health: Proceedings of the 15th World Congress on Health and Biomedical Informatics 216* (2015).

[86] Rodrigues, E. O.; Pinheiro, V. H. A.; Liatsis, P.; Conci, A. Machine learning in the prediction of cardiac epicardial and mediastinal fat volumes. *Computers in Biology and Medicine* (2017).

[87] Rodrigues, E. O.; Porcino, T. M.; Conci, A.; Silva, A. C. A simple approach for biometrics: Finger-knuckle prints recognition based on a sobel filter and similarity measures. *International Conference on Systems, Signals and Image Processing (IWSSIP)* (2016).

[88] Rodrigues, E. O.; Rodrigues, L. O.; Oliveira, L. S. N.; Conci, A.; Liatsis, P. Automated recognition of the pericardium contour on processed ct images using genetic algorithms. *Computers in Biology and Medicine 87* (2017), 38–45.

[89] Rodrigues, E. O.; Torok, L.; Liatsis, P.; Viterbo, J.; Conci, A. k-ms: A novel clustering algorithm based on morphological reconstruction. *Pattern Recognition 66* (2016), 392–403.

[90] Rodrigues, E. O.; Viterbo, J.; Conci, A.; McHenry, T. A context-aware middleware for medical image based reports an approach based on image feature extraction and association rules. *IEEE International Conference on Computer Systems and Applications* (2015).

[91] Rosenblatt, F. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Spartan Books, 1961.

[92] Russell, S.; Norvig, P. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.

[93] Serna, A.; Marcotegui, B. Detection, segmentation and classification of 3d urban objects using mathematical morphology and supervised learning. *ISPRS Journal of Photogrammetry and Remote Sensing 93* (2014), 243–255.

[94] Sharma, Y.; Meghrajani, Y. K. Brain tumor extraction from mri image using mathematical morphological reconstruction. *Emerging Technology Trends in Electronics, Communication and Networking* (2014).

[95] Shattuck, D. W.; Sandor-Leahy, S. R.; Schaper, K. A.; Rottenberg, D. A.; Leahy, R. M. Magnetic resonance image tissue classification using a partial volume model. *NeuroImage 13* (2001), 856–876.

[96] Silva, L. F.; Santos, A.; Bravo, R. S.; Silva, A. C.; Muchaluat-Saade, D. C.; Conci, A. Hybrid analysis for indicating patients with breast cancer using temperature time series. *Computer Methods and Programs in Biomedicine 130* (2016), 142–153.

[97] Silva, L. F.; Santos, A. A.; Bravo, R. S.; Silva, A. C.; Muchaluat-Saade, D. C.; Conci, A. Hybrid analysis for indicating patients with breast cancer using temperature time series. *Computer Methods and Programs in Biomedicine 130* (2016), 142–153.

[98] Sonka, M.; Hlavac, V.; Boyle, R. *Image Processing, Analysis and Machine Vision.* Springer US, 1993.

[99] Sonka, M.; Hlavac, V.; Boyle, R. *Image Processing, Analysis and Machine Vision.* Cengage Learning, 2014.

[100] Spyromitros, E.; Tsoumakas, G.; Vlahavas, I. An empirical study of lazy multilabel classification algorithms. *Artificial Intelligence: Theories, Models and Applications 5138* (2008), 401–406.

[101] Stuart, R.; Peter, N. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2003.

[102] Su, M.; Chou, C. A modified version of the k-means algorithm with a distance based on cluster symmetry. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23* (2001), 674–680.

[103] Thornton, C.; Hutter, F.; Hoos, H. H.; Leyton-Brown, K. Auto-weka: combined selection and hyperparameter optimization of classification algorithms. *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2013), 847–855.

[104] Torok, L.; P., M.; Trevisan, D. G.; Clua, E.; Montenegro, A. A mobile game controller adapted to the gameplay and users behavior using machine learning. *Entertainment Computing - ICEC 9353* (2015), 3–16.

[105] Torok, L.; Pelegrino, M.; Trevisan, D.; Clua, E.; Montenegro, A. A mobile game controller adapted to gameplay and user's behavior using machine learning. *Entertainment Computing - ICEC 2015 9353* (2015), 3–16.

[106] Tuia, D.; Pacifici, F.; Kanevski, M.; Emery, W. J. Classification of very high spatial resolution imagery using mathematical morphology and support vector machines. *IEEE Transactions on Geoscience and Remote Sensing 17* (2009), 3866–3879.

[107] UCI. Uci datasets. http://repository.seasr.org/Datasets/UCI/arff/, 2016.

[108] Vincent, L. Morphological grayscale reconstruction in image analysis: applications and efficient algorithms. *Image Processing 2* (1993), 176–201.

[109] Wang, L.; Zhang, Y.; Feng, J. On the euclidean distance of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 8 (2005), 1334–1339.

[110] Wang, M.; Zhou, C.; Pei, T.; Luo, J. A mathematical morphology based scale space method for the mining of linear features in geographic data. *Data Mining and Knowledge Discovery 12* (2006), 97–118.

[111] Weinberger, K. Q.; Saul, L. K. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research 10* (2009), 207–244.

[112] Witten, T. A.; Sander, L. M. Diffusion-limited aggregation, a kinetic critical phenomenon. *Physical Review Letters 47*, 19 (1981).

[113] Xiu, R.; Wunsch, D. Survey of clustering algorithms. *IEEE Transactions on Neural Networks 16* (2005), 645–678.

[114] Yousri, N. A.; Kamel, M. S.; Ismail, M. A. A distance-relatedness dynamic model for clustering high dimensional data of arbitrary shapes and densities. *Pattern Recognition 42* (2009), 1193–1209.

[115] Yu, J.; Hong, R.; Wang, M.; You, J. Image clustering based on sparse patch alignment framework. *Pattern Recognition 47* (2014), 3512–3519.

[116] Zarinbal, M.; Zarandi, M. H. F.; Turksen, I. Relative entropy collaborative fuzzy clustering method. *Pattern Recognition 48* (2015), 933–940.

[117] Ze-Feng, D.; Zhou-Ping, Y.; You-Lun, X. High probability impulse noise-removing algorithm based on mathematical morphology. *IEEE Signal Processing Letters* (2006), 31–34.

[118] Zhong, C.; Yue, X.; Zhang, Z.; Lei, J. A clustering ensemble: Two-level-refined co-association matrix with path-based transformation. *Pattern Recognition 48* (2015), 2699–2709.

[119] Zhou, Y.; Wang, R.; Huang, W.; Yuan, Y.; Chen, X.; Wang, L. Attenuation of diffraction noise in marine surveys with mathematical morphology. *Society of Exploration Geophysicists* (2016), 5654.

[120] Zhu, Z.; Shen, J.; Yu, H. Noise removal and fracture analysis in borehole images using mathematical morphology and compressive sensing. *78th EAGE Conference and Exhibition 2016* (2016).