UNIVERSIDAD DE OVIEDO

MASTER IN SOFT COMPUTING
AND INTELLIGENT DATA ANALYSIS

PROYECTO FIN DE MASTER
MASTER PROJECT

# IMPROVING THE READABILITY OF DOCUMENTS BY CUSTOMIZING THE MOST RELEVANT FIGURES ON THE FLY

RUBÉN GONZÁLEZ TURANZAS
JULY 2012

UNIVERSIDAD DE OVIEDO

MASTER IN SOFT COMPUTING
AND INTELLIGENT DATA ANALYSIS

PROYECTO FIN DE MASTER
MASTER PROJECT

# IMPROVING THE READABILITY OF DOCUMENTS BY CUSTOMIZING THE MOST RELEVANT FIGURES ON THE FLY

RUBÉN GONZÁLEZ TURANZAS

JULY 2012

TUTOR / ADVISOR:
JOSÉ OTERO RODRIGUEZ

# Index

# 1. Summary and keywords

This project was motivated for the increasing use of the small screen devices, such as smartphones, and the fact that technological advances seem to lead to a future where the digital contents will dominate over other media.

Reading contents in small screen devices has drawbacks related with visual capabilities, size and reading comprehension. Inside this work, references for papers in those subjects can be found but there are no works that study the whole problem together.

Probably because it is a challenge not tackled yet, there isn't a practical solution which can improve the readability of documents by customizing them depending on the visual capabilities of the user.

The proposal of this work will face the challenge using different soft computing techniques of clustering in order to compare results. Hierarchical agglomerative, K-means, multivariate Gaussian mixture models and fuzzy K-means clustering, with several variations, will be experimented in eleven text documents.

The datasets obtained from the documents were extracted following a methodology that exploits techniques of mathematical morphology and statistics. Accordingly, the data will be clustered to three classes that symbolise the three basic zooms for the reading process: standard zoom, zoom for small element and zoom for large elements, all customizable by the user.

The percentage of success reached for some of the techniques regarding the clustered error rate is quite high (mean near to 84%) and encouraging this researcher to continue working in the future with practical applications in the digital devices involved in the current daily life.

**Keywords:** readability, text, clustering, computer vision, screens.

# 2. Intoduction

The use of small electronic devices such as smartphones, tablets or eBooks readers to read documents has been increasing significantly in the last years. Also, it is postulated to be the trend of the future because some devices, like mobile phones, would always be small by concept. Furthermore, the technology advancements usually go in the direction of ‘more in less space’.

The evolution of contents such as websites, emails, eBooks, downloaded text documents or instructions to follow in diverse situations, is a constant and it seems to expand more and more.

If we consider a text document as a product, it doesn’t matter the media in which it is performed (paper, computer screens or small devices screens). Furthermore, if we consider the person who reads any kind of text document as a consumer we could say that the vast majority of the industry which “manufacture” text documents (from the public administration to the private company, going through all the possibilities like advertising, entertainment, or education among others) is focused on paper and medium/big screens. There are a very small percentage of documents aimed to the small screens devices. As it was said before, it is very probable that the contents to the small screen devices would increase with time, but those three main media will have to coexist for a long time.

From the consumer’s experience viewpoint, the relation between computer screens and paper has improved in the last years due to the increase of the size of the standard screens and the pixels per inch rate (ppi). Those two improvements make it possible to view the text pages almost at the same size of an A4 page. If the software aids developed so far are also considered, the comfort of using digital media is most of the times superior to the paper media. We can remark the inconveniences that the paper support has like storing, searching, sharing or editing. On the other hand, it has provided a great advantage so far as it is a fact that reading from paper causes less eye fatigue that reading directly from screens.

The relation between computer screens and small screen devices is a matter of size, understanding the small screens as screens smaller than 7’’. The common digital contents which, few years ago, were exclusively aimed to computers are being visualized now in small devices too, so the reading process is executed by the consumer due to the possibility of using different techniques of zooming and scrolling. Depending on the size involved in the fonts, images or pages, and the distances between lines, characters or columns, reading can be more comfortable for the user or a struggling experience to adjust the contents to the visual capabilities.

The classical technique of vertical scrolling is assumed for the user as a necessary basic function as it happens in the large screens but generally, in small screens, horizontal repetitive scrolling is needed too. We can add the fact that scrolling in small screen devices such as smartphones is done using the touch screen technology so the scrolling is not executed perfectly horizontal or vertical. Also zooming is essential in small screens. Those effects make a loss of concentration in the reading and goes in decrease of the quality in terms of text understanding or assimilating.

There are different techniques for zooming but the most popular is the "pinch open & close zooming" where the user has to place two fingers in the screen and drag them near each other to open the zoom and drag them far from each other to close the zoom. This technique has a collateral effect: easy and unintentional scrolling is also performed. When all of these negative effects are put together, the reading experience is not comfortable and loses quality drastically.

There are some researches done on the different factors commented above but there is no one about all of them together in the whole user experience. There is no research on a possible solution nor any kind of application developed so far.

This master project carries out a research in the different techniques that can drive to a possible future application. The aim was to propose some methodologies or solutions for the described problem above and the different difficulties that could hide within, using several of the wide range of soft computing techniques studied in the master during the year. The comparison between them and the rejection of the bad approaches is a part of the research, as it will be detailed later.

# 3. State of the art

In order to structure all the information gathered about the subject, the main questions are going to be organized in five main points:

a) Present technology and small computer devices.
   Notebooks, tablets, smartphones, etc...

b) Interacting tools and methods for computer devices.

c) The influence of size in the reading understanding

d) Achievements on image segmentation and text clustering applied to reading documents

e) Text and image blocks sorting on reading documents

## 3.1. Present technology and small computer devices. Notebooks, tablets, smartphones, etc...

The visualization through electronic devices has been an important subject ~~to~~ of research and development since the first computers appeared. It has been in constant evolution together with the hardware and the software involved in the matter.

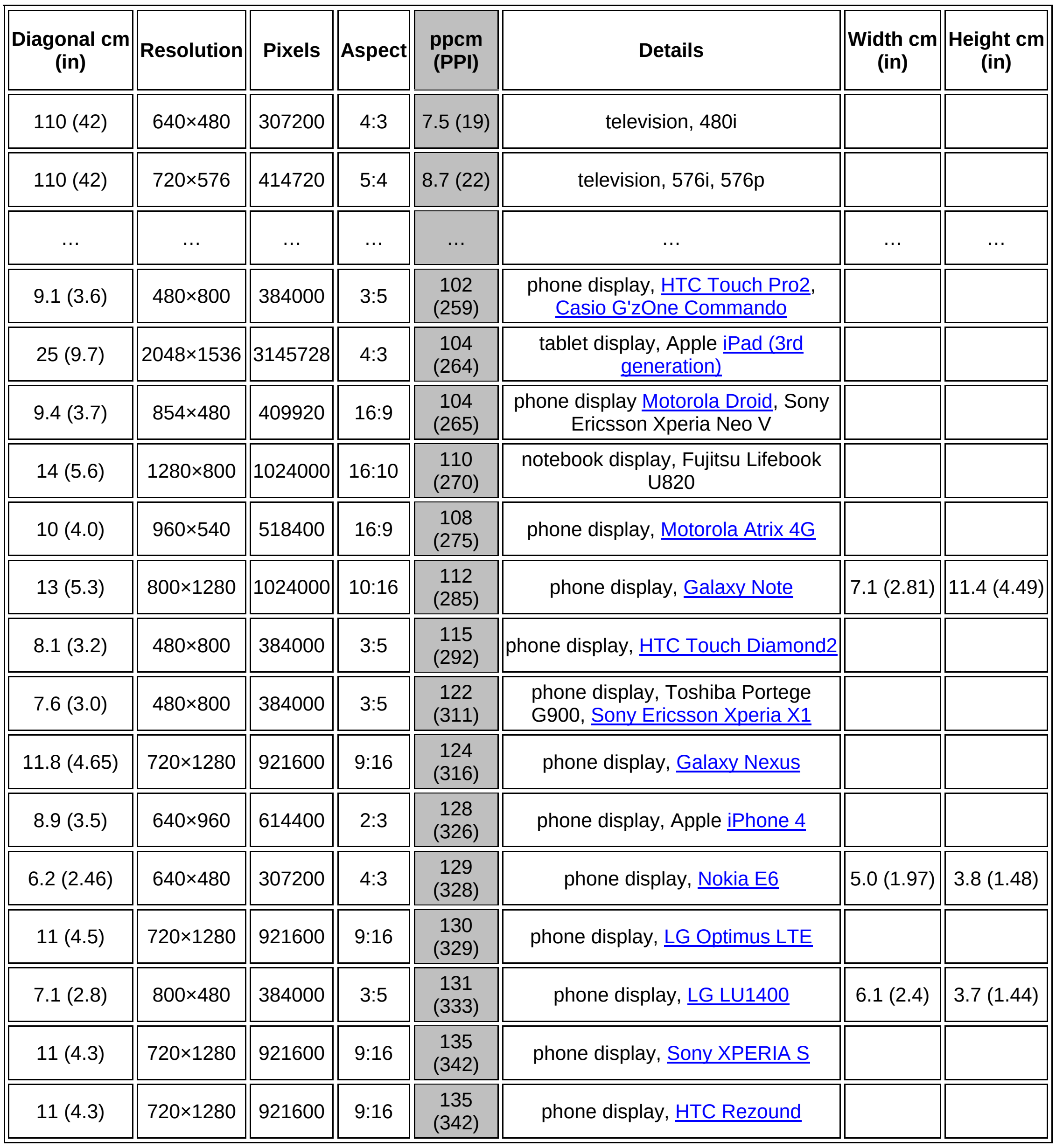
The first generation of devices designed to visualize the information of a computer was the CRT monitors (cathode ray tube). At the beginning they were analogical and monochromatic voluminous screens which evolved to the actual generation of digital realistic-color ultra-slim screens. The first commercial computer screens had very low resolution as, for instance, 160x200 pixels. Now we can reach ultrahigh resolutions like 7680 × 4320 pixels [16]. At the same time the pixel size has become smaller to the point that the human eye can not notice it at a standard distance of use. The measurement for this characteristic is dpi (dots per inch) or ppi (pixels per inch). The first commercial PC had 72 dpi or less, while now we have screens with around 500 dpi at the reach of the common user.

At the present, the improvement of the dpi is an important point in R+D+I for the telecommunication device manufacturers because of the tendency of having more powerful electronic creations in smaller devices. As before, the Personal Computers trend tries to have more powerful machines in the smallest volume, the laptops followed them and nowadays the next achievement seems to have the most powerful computer in a similar mobile phone size.

Recently, a new kind of electronic devices came up from the manufactures and is taking a place in the family of the common work/home computer devices. They are the tablets, placed in an intermediate position between the mobile phones and the laptops. It is also remarkable the emergence of tiny laptops with around 8'' or 10'' screen sizes These laptops are extremely light and are usually referred as notebooks, ultrabooks, netbooks,  eBook readers, etc.

For our work, we will have more consideration on the mobile phones because they are the devices that look for the most powerful characteristics (as the rest of the mobile computers) but with the smallest size. They are called smartphones.

Nowadays, we have a wide range of devices with different dpi measurements. Here is part of a list that can be consulted in [1]. Devices sorted by ppi:



| Diagonal cm (in) | Resolution | Pixels | Aspect | ppcm (PPI) | Details | Width cm (in) | Height cm (in) |
|---|---|---|---|---|---|---|---|
| 110 (42) | 640×480 | 307200 | 4:3 | 7.5 (19) | television, 480i | | |
| 110 (42) | 720×576 | 414720 | 5:4 | 8.7 (22) | television, 576i, 576p | | |
| … | … | … | … | … | … | … | … |
| 9.1 (3.6) | 480×800 | 384000 | 3:5 | 102 (259) | phone display, HTC Touch Pro2, Casio G'zOne Commando | | |
| 25 (9.7) | 2048×1536 | 3145728 | 4:3 | 104 (264) | tablet display, Apple iPad (3rd generation) | | |
| 9.4 (3.7) | 854×480 | 409920 | 16:9 | 104 (265) | phone display Motorola Droid, Sony Ericsson Xperia Neo V | | |
| 14 (5.6) | 1280×800 | 1024000 | 16:10 | 110 (270) | notebook display, Fujitsu Lifebook U820 | | |
| 10 (4.0) | 960×540 | 518400 | 16:9 | 108 (275) | phone display, Motorola Atrix 4G | | |
| 13 (5.3) | 800×1280 | 1024000 | 10:16 | 112 (285) | phone display, Galaxy Note | 7.1 (2.81) | 11.4 (4.49) |
| 8.1 (3.2) | 480×800 | 384000 | 3:5 | 115 (292) | phone display, HTC Touch Diamond2 | | |
| 7.6 (3.0) | 480×800 | 384000 | 3:5 | 122 (311) | phone display, Toshiba Portege G900, Sony Ericsson Xperia X1 | | |
| 11.8 (4.65) | 720×1280 | 921600 | 9:16 | 124 (316) | phone display, Galaxy Nexus | | |
| 8.9 (3.5) | 640×960 | 614400 | 2:3 | 128 (326) | phone display, Apple iPhone 4 | | |
| 6.2 (2.46) | 640×480 | 307200 | 4:3 | 129 (328) | phone display, Nokia E6 | 5.0 (1.97) | 3.8 (1.48) |
| 11 (4.5) | 720×1280 | 921600 | 9:16 | 130 (329) | phone display, LG Optimus LTE | | |
| 7.1 (2.8) | 800×480 | 384000 | 3:5 | 131 (333) | phone display, LG LU1400 | 6.1 (2.4) | 3.7 (1.44) |
| 11 (4.3) | 720×1280 | 921600 | 9:16 | 135 (342) | phone display, Sony XPERIA S | | |
| 11 (4.3) | 720×1280 | 921600 | 9:16 | 135 (342) | phone display, HTC Rezound | | |

*Figure 1 – List of displays by pixel density*

## 3.2 Interacting tools and methods for computer devices.

The visualization of text documents in screens can be easier and more comfortable if the size of the screen is bigger and the resolution of the image also is increased. Thus, it improves the hardware aspect of the subject, but there are other ways relative to the software aspect that can also be improved as we will see.

Regarding the hardware aspect, although the number of ppi (pixels per inch) and the size of the screens have increased, the contents are usually larger than the screen capability of the small device. Consequently, there is a need to work out the situation and navigate through all the information.

Considering the software aspect of the subject, when the mobile devices became more and more popular, a new programming standard ~~were~~ was created specifically to display contents in those small screens. It was WAP (Wireless Application Protocol), an open standard for application in wireless communications. When wap was developed, the screens, their ppi and the data transfer speed of the nets were much poorer, so the contents had to be adapted to something much lighter and schematic.

In the present time, technology allows to have good data transfer rates so the improvements in screen sizes and ppi density have reduced the difference in the web programming and wap programming. It probably would disappear in a future, so the tendency now in the smartphones is to create tools to navigate in universal contents, not just in wap contents.

The most remarkable tools aimed to aid the visualization are:

### *3.2.1 Double scroll bar, vertical and horizontal.*

This is the traditional way to navigate horizontally or vertically through the contents.

*Figure 2 – Horizontal scrolling bar*

### *3.2.2 Zoom buttons*

It depends on the software and on the operative system, and if they do incremental or decreasing fixed zoom in the contents.

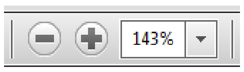


*Figure 3 – Typical Zoom buttons*

### *3.2.3 Two-level zoom [2]*

A variation from the previous tool is the two-level zoom. It works with two zooms, one zoom in and one zoom out (or back to previous zoom). It is usually found in the touch-screens of the smartphones where the default zoom shows the general content. A

## 3.2 Interacting tools and methods for computer devices.

The visualization of text documents in screens can be easier and more comfortable if the size of the screen is bigger and the resolution of the image also is increased. Thus, it improves the hardware aspect of the subject, but there are other ways relative to the software aspect that can also be improved as we will see.

Regarding the hardware aspect, although the number of ppi (pixels per inch) and the size of the screens have increased, the contents are usually larger than the screen capability of the small device. Consequently, there is a need to work out the situation and navigate through all the information.

Considering the software aspect of the subject, when the mobile devices became more and more popular, a new programming standard ~~were~~ was created specifically to display contents in those small screens. It was WAP (Wireless Application Protocol), an open standard for application in wireless communications. When wap was developed, the screens, their ppi and the data transfer speed of the nets were much poorer, so the contents had to be adapted to something much lighter and schematic.

In the present time, technology allows to have good data transfer rates so the improvements in screen sizes and ppi density have reduced the difference in the web programming and wap programming. It probably would disappear in a future, so the tendency now in the smartphones is to create tools to navigate in universal contents, not just in wap contents.

The most remarkable tools aimed to aid the visualization are:

### 3.2.1 Double scroll bar, vertical and horizontal.

This is the traditional way to navigate horizontally or vertically through the contents.

*Figure 2 – Horizontal scrolling bar*

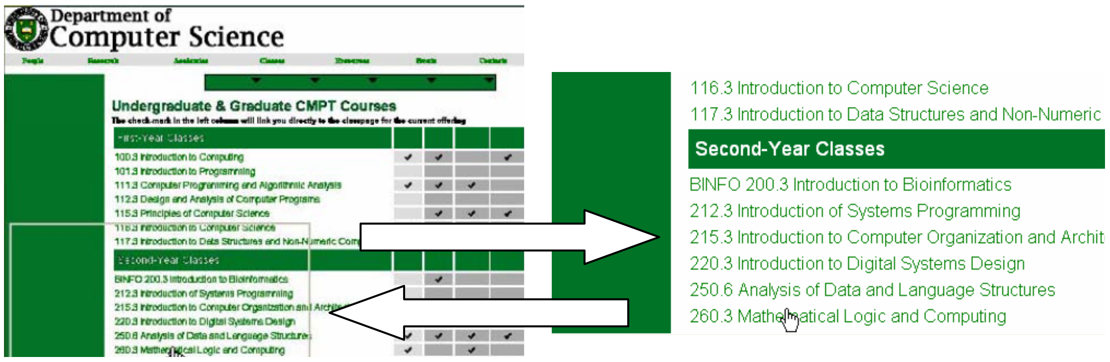
### 3.2.2 Zoom buttons

It depends on the software and on the operative system, and if they do incremental or decreasing fixed zoom in the contents.

*Figure 3 – Typical Zoom buttons*

### 3.2.3 Two-level zoom [2]

A variation from the previous tool is the two-level zoom. It works with two zooms, one zoom in and one zoom out (or back to previous zoom). It is usually found in the touch-screens of the smartphones where the default zoom shows the general content. A quick double touch in the screen does a prefixed increment of the zoom in and another quick double touch returns to the previous zoom sight.


Department of
Computer Science

Undergraduate & Graduate CMPT Courses
The check-mark in the left column will link you directly to the classpage for the current offering

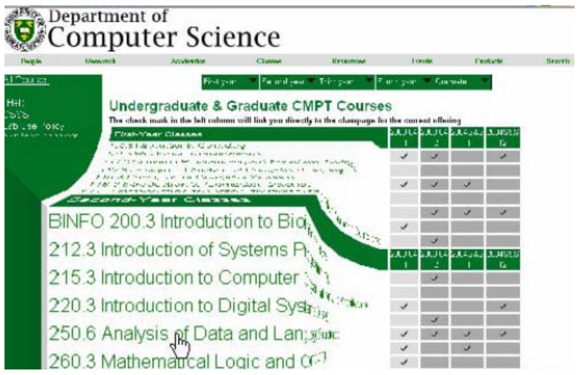
First-Year Classes
100.3 Introduction to Computing
101.3 Introduction to Programming
111.3 Computer Programming and Algorithmic Analysis
112.3 Design and Analysis of Computer Programs
115.3 Principles of Computer Science
116.3 Introduction to Computer Science
117.3 Introduction to Data Structures and Non-Numeric

Second-Year Classes
BINFO 200.3 Introduction to Bioinformatics
212.3 Introduction of Systems Programming
215.3 Introduction to Computer Organization and Archit
220.3 Introduction to Digital Systems Design
250.6 Analysis of Data and Language Structures
260.3 Mathematical Logic and Computing

Figure 4 – Two level zoom example

### 3.2.4 Fisheye [2]

Another variation of the previous tool is the fisheye. It divides the screen in three concentric regions around the position of the pointer. The inner region experiments a zoom in. The outer region keeps the zoom as default. The middle region does a transition from the inner to the outer region. It had no success in mobile devices so it is no longer used.


Department of
Computer Science

Undergraduate & Graduate CMPT Courses

BINFO 200.3 Introduction to Bio
212.3 Introduction of Systems P
215.3 Introduction to Computer
220.3 Introduction to Digital Sys
250.6 Analysis of Data and Lan
260.3 Mathematical Logic and C

Figure 5 – Fisheye example

### 3.2.5 Rapid serial visual presentation (RSVP)

It is a method of displaying information (generally text or images) in which the text is displayed word-by-word in a fixed focal position. Aside from a basic reading aid, RSVP is being researched as a tool to increase individual reading rates. RSVP is also being used for research in the fields of visual impairment, dyslexia, perceptual and cognitive psychology. It had no success in mobile devices and nowadays its use would make no sense with the latest technology advances available.

### *3.2.6 RotoView complements touch-screens [3]*

RotoView is a tilt-based tool thought for devices that can notice its own inclination with the aim of controlling the view navigation. RotoView enables the user to navigate with only one hand. The idea is simple, it simulates the content is a sliding plate which self scrolls depending on the device inclination that the user performs.

### *3.2.7 Mac finger gestures [4]*

Mac was the company that made them famous for the touch-screens, and they had a great success. Their team has developed many gestures but the most important ones for navigating are two:

#### *3.2.7.1 Grab & Drag.*

In order to do horizontal scrolling, vertical scrolling or both at the same time; the user only has to touch the screen, holding the finger in contact with the screen and moving it around.

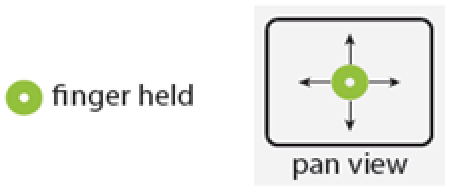


*Figure 6 – Pan view gesture*

#### *3.2.7.2 Pinch open & close.*

Here two fingers are needed. The user has to place two fingers in two points of the screen and hold them touching. The content keeps focused on the center of the imaginary line of the two finger touching points. The zoom effect is performed when the two finger touching points get closer (zoom out) or further (zoom in).

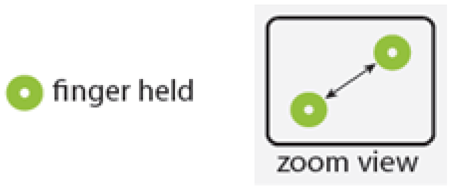


*Figure 7 – Zoom view gesture*

### *3.2.8 Leap motion [5]*

It is the newest tool to control the navigation (apart from other applications). It probably would be the next evolution step for controlling computer devices, although for smartphones it could not be the best practical solution.

It’s more accurate than a mouse, as reliable as a keyboard and more sensitive than a touch-screen. For the first time, you can control a computer in three dimensions with your natural hand and finger movements.

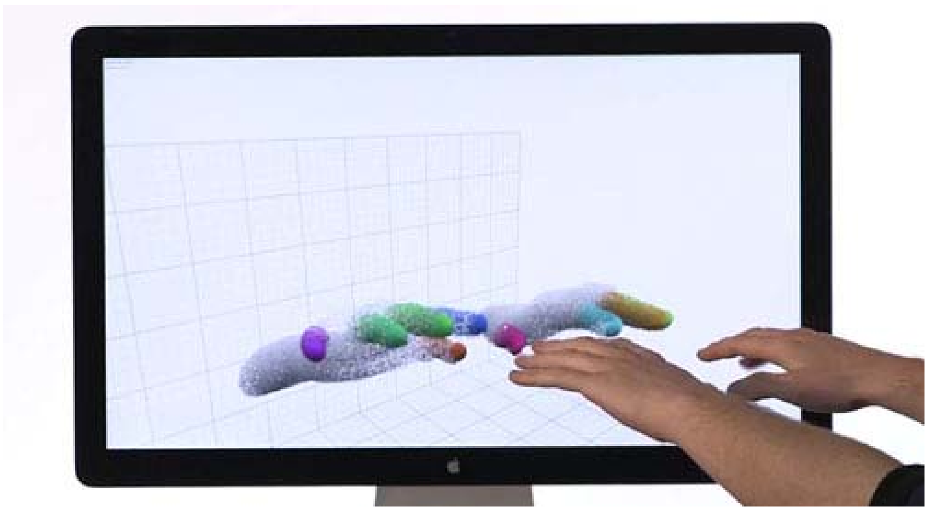


*Figure 8 – Leap motion*

## 3.3 The influence of size in the reading understanding

We can read in several research works that the size is a factor that matters for the compressibility of the contents displayed in the small screen. It has influenced in different ways:

### *3.3.1 The influence of the size of the screen*

Dillon et al. [6] presented experiment based on a 3,500 word text, using a 20 and 60 line display window. Subjects were asked to read the texts and later summarize the main points. The study found that the comprehension rates on the smaller screen were as good as those on the larger one.

Dillon's study highlighted other two effects of smaller displays. Firstly, users reading from the small displays interacted with the display window to a much higher degree than those with the larger window. Users paged much more backwards and forwards through the text in the small screen display. They may have done this in an attempt to orientate themselves, to provide them with context as they progressed through the text. Web pages, of course, usually contain far more structure than the texts presented by Dillon and others. We might expect, then, handheld Web browser users to make significant use of any scrolling and paging mechanisms in order to help them make sense of the page.

Secondly, Dillon also found that 75% of users who indicated they would have liked to have changed their screen size had used the small screen display. This suggests, perhaps, that even if objectively performance is not affected, first-time small screen Web users might perceive the systems as being less good than the conventional platforms.

#### 3.3.2 The influence of the size of the width-line

In an early study, Duchnicky and Kolers [7] considered the effect on reading of window height and line widths. The full width display was read 25% faster than the screen which was 1/3 the width. However, the impact of varying the display height was very much less dramatic.

Although very small window sizes (1–2 lines) gave poor performance, the optimal height was found to be just 4 lines. There were no significant improvements in comprehension when the display height was increased to 20.

#### 3.3.3 The influence of the density of characters per line

Duchnicky and Kolers [7] studied the readability of text scrolled on visual display terminals as a function of two different character densities. Text appearing at a density of 80 characters per line was read 30% faster than text in a format of 40 characters per line. Text appearing in windows four lines high was read as efficiently as text in 20-line window, and text in one or two-line windows was read only 9% more slowly than text in 20-line window. Comprehension of the passages did not vary as a function of window size, indicating that subjects maintained a constant level of comprehension by varying their reading rate.

#### 3.3.4 The influence of the size of the characters

Though there have been many studies of computer based text reading, only a few have considered the small screens of handheld computers. The paper of Darroch et al. presents an investigation into the effect of varying font size between 2 and 16 point on reading text on a handheld computer. By using both older and younger participants the possible effects of age were examined. Reading speed and accuracy were measured and subjective views of participants recorded. Objective results showed that there was little difference in reading performance above 6 point, but subjective comments from participants showed a preference for sizes in the middle range. We therefore suggest, for reading tasks, that designers of interfaces for mobile computers should provide fonts in the range of 8-12 point to maximize readability for the widest range of users.

#### 3.3.5 The influence of the quantity of the scrolling needed

Kim and Albers [9] investigated the effect of table formatting on a small-screen interface, and they found that horizontal scrolling could affect users' information searching performance.

This study explored strengths and limitations of table formatting choices by engaging twenty-eight participants in information searches in online tables, presented on a small-screen interface (Palm IIIc). Table length across conditions was held constant at three screens long (24 rows total) but it varied from one to three screens wide (approximately 35, 70, and 105 characters per line). Target information was positioned in either the upper left, lower left, upper right, or lower right quadrants. Data collected were time on task, error rate, and level of participants' confidence in their answers. Researchers found that increased horizontal scrolling imposed the heaviest burden on information search. This study supports restricting table widths to one screen on handheld computers. If necessary, however, tables can go to two screens wide without critical detriment to usability. While ruled line formatting is slightly better than interface character in providing visual support for the burden of horizontal scrolling, neither formatting option adequately compensates for the added burden.

## 3.4 Achievements on image segmentation and text clustering applied to reading documents

One of the most remarkable works done in this particular field was done by Caponetti et al. [10] that propose a new document page segmentation method which can differentiate between text, graphics and background, using a neuro-fuzzy methodology. Their approach is based firstly on the analysis of a set of features extracted from the image, available at different resolution levels. An initial segmentation is obtained by classifying the pixels into coherent regions, which are successively refined by the analysis of their shape. The core of the approach relies on a neuro-fuzzy methodology, to perform the classification processes. The proposed strategy can describe the physical structure of a page in an accurate way and it proved to be robust against noise and page skew. Additionally, the knowledge-based neuro-fuzzy methodology allows us to understand the classification mechanisms in a better way, contrary to what happens when other kinds of knowledge-free methods are applied.

Another outstanding work was the research done by Yen-Lin Chen et al. [11] that presented a new knowledge-based system for extracting and identifying text-lines from various real-life mixed text/graphics compound document images. The proposed system first decomposes the document image into distinct object planes to separate homogeneous objects, including textual regions of interest, non-text objects such as graphics and pictures, and background textures. A knowledge-based text extraction and identification method obtains the text-lines with different characteristics in each plane. The proposed system offers high flexibility and expandability by merely updating new rules to cope with various types of real-life complex document images. Experimental and comparative results prove the effectiveness of the proposed knowledge-based system and its advantages in extracting text lines with a large variety of illumination levels, sizes, and font styles from various types of mixed and overlapping text/graphics complex compound document images.

Jie Xi et al. [12] also did a great work of text pages segmentation based on run-length smoothing algorithm and minimal spanning tree clustering. The proposed method can solve the problems of segmenting Chinese documents that differ from English documents too.

Going a step further, Apostolos Antonacopoulos [13] did an interesting work in which he develops a method able to segmentate pages containing nonrectangular text regions interleaved with images.

## 3.5 Text and image blocks sorting on reading documents

Probably the most remarkable work that has been made in text or image sorting is done for digital comic or manga, where the frames need to be isolated from the rest of the page and sorted to follow the natural reading line.

Yamada et al. [14] did one of the most advanced works done so far. Their algorithm is able to isolate frames with irregular polygonal shapes. It is based on a simple concept but quite complex to implement. They create lines between the different frames previously isolated, and by analyzing them and their borders, the algorithm is able to determine the sequence to follow.

# 4. Goals and personal approach

The initial goal of this project was a practical solution to small screen devices, more specifically a software application where the user could customize, according to his/her own visual capabilities, the visual properties of a document text. The application would help him/her to be focused on the mental process of reading and understanding the text, and not in the constant scrolling and zooming of the screen. Those two functions would be reduced to the minimum by a previous preprocess that the application would execute to the document.

After spending an important part of the project studying, discovering all the challenges hidden within, researching and achieving all the small goals to get near of the final solution; I realized that the task will need a longer project time. Consequently, my project only tackles the solution of the basis challenges related with soft computing techniques.

The whole concept of the application involves several problems that need to be solved a priori. The first situation to be solved is the extraction of the document visual information to a dataset of characteristics. Then, different techniques of soft computing can be applied to the data.

In order to avoid the drawbacks derived from the extraction of the visual information such as the implementation of different text reading software (pdf readers, text editors, web browsers, ...) or the programming in different operative systems (windows, Mac, Linux, ...), I decided to read the documents as an external device or human would do it. The information would be read directly from the screen, as a sum of ordered pixels.

For example, it is possible to analyze the html code of a web page (probably the more accessible and less opaque codification) or we can analyze the drawn pixels of the screen in a specific moment. The second option seems to be a greatest challenge but if it is achieved, it would retrieve us much more information to the project goal.

The solution proposed is universal to every device, operative system, software or digital support. Everything with a screen made from a structure of pixels would support this application. Another benefit is the possibility to work with scanned texts because they are already a sum of ordered pixels.

Thus, the documents have to be converted previously to images and then, analyzed by the application. Due to the (a priori) less powerful capabilities of the small devices, the soft computing techniques used will have to suppose low computing cost.

The process followed to create the dataset from a text document is enumerated in the next steps:

- Find some documents with different fonts, several graphics, tables and images that will need repetitive changes of zoom and massive scrolling.
- Extract the areas of information from the background. Dilation
- Extract the standard interline space, the standard intercolumn space and the height of the standard line of the document. Dilation kernel.
- Find the optimal image resolution to work with. Downsampling.
- Compare between smoothed and not smoothed image downsampling.
- Binarize the dilated image.
- Label not connected areas.
- Fill the holes of the labeled areas.
- Joint related areas of information.
- Sort roughly of the information areas (i.e. sorting of paragraphs and columns)
- Sort finely the internal elements of each rough area (i.e. sorting lines within a paragraph)

Once the dataset is created, the following steps are involved with preprocessing:

- Determine the space of characteristics
- Describe the metrics used
- Set the input space
- Set the output space

Finally, four different clustering techniques (and some of their variants) are going to be applied and their results compared. They are agglomerative hierarchical cluster, k-means clustering, multivariate Gaussian mixture clustering and fuzzy k-means clustering.

# 5 Methodology and theoretical fundamentals

The experiments carried out were based on eleven text documents chosen properly to the project. They are guided through a methodology that goes from the election of the examples, covering all the necessary goals to achieve the final clustering.

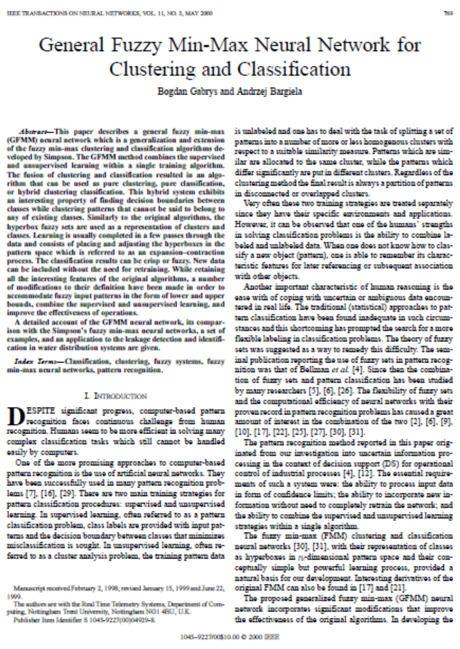
## 5.1 Examples or Datasets chosen

The main goal in this point is finding documents with different fonts, several graphics, tables and images that will need repetitive changes of zoom and massive scrolling.

The eleven chosen text documents are:

Document 1 [18], document 2 [19], document 3 [20], document 4 [21], document 5 [22], document 6 [23], document 7 [24], document 8 [25], document 9 [26], document 10 [27], document 11 [28].

In order to guide the experiments, and because it is the example that best represents the previous concepts, the document number 1 is going to be chosen as the reference document for my work in this project. It is possible to see in fig. 9 the different sizes of fonts including the formulas, the sorting confusion that can suppose the open graphics, formulas or schemas (meaning open as not embedded in a box or similar). The fact of having interleaved sections with two columns format and one column format could propitiate sorting errors. Also, it is remarkable that some images are fitted into the text.



IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 11, NO. 3, MAY 2000 769

General Fuzzy Min-Max Neural Network for Clustering and Classification

Bogdan Gabrys and Andrzej Bargiela

***Abstract*—This paper describes a general fuzzy min-max (GFMM) neural network which is a generalization and extension of the fuzzy min-max clustering and classification algorithms developed by Simpson. The GFMM method combines the supervised and unsupervised learning within a single training algorithm. The fusion of clustering and classification resulted in an algorithm that can be used as pure clustering, pure classification, or hybrid clustering classification. This hybrid system exhibits an interesting property of finding decision boundaries between classes while clustering patterns that cannot be said to belong to any of existing classes. Similarly to the original algorithms, the hyperbox fuzzy sets are used as a representation of clusters and classes. Learning is usually completed in a few passes through the data and consists of placing and adjusting the hyperboxes in the pattern space which is referred to as an expansion–contraction process. The classification results can be crisp or fuzzy. New data can be included without the need for retraining. While retaining all the interesting features of the original algorithms, a number of modifications to their definition have been made in order to accommodate fuzzy input patterns in the form of lower and upper bounds, combine the supervised and unsupervised learning, and improve the effectiveness of operations.**

**A detailed account of the GFMM neural network, its comparison with the Simpson's fuzzy min-max neural networks, a set of examples, and an application to the leakage detection and identification in water distribution systems are given.**

***Index Terms*—Classification, clustering, fuzzy systems, fuzzy min-max neural networks, pattern recognition.**

I. INTRODUCTION

DESPITE significant progress, computer-based pattern recognition faces continuous challenge from human recognition. Humans seem to be more efficient in solving many complex classification tasks which still cannot be handled easily by computers.

One of the more promising approaches to computer-based pattern recognition is the use of artificial neural networks. They have been successfully used in many pattern recognition problems [7], [16], [29]. There are two main training strategies for pattern classification procedures: supervised and unsupervised learning. In supervised learning, often referred to as a pattern classification problem, class labels are provided with input patterns and the decision boundary between classes that minimizes misclassification is sought. In unsupervised learning, often referred to as a cluster analysis problem, the training pattern data is unlabeled and one has to deal with the task of splitting a set of patterns into a number of more or less homogenous clusters with respect to a suitable similarity measure. Patterns which are similar are allocated to the same cluster, while the patterns which differ significantly are put in different clusters. Regardless of the clustering method the final result is always a partition of patterns in disconnected or overlapped clusters.

Very often these two training strategies are treated separately since they have their specific environments and applications. However, it can be observed that one of the humans' strengths in solving classification problems is the ability to combine labeled and unlabeled data. When one does not know how to classify a new object (pattern), one is able to remember its characteristic features for later referencing or subsequent association with other objects.

Another important characteristic of human reasoning is the ease with of coping with uncertain or ambiguous data encountered in real life. The traditional (statistical) approaches to pattern classification have been found inadequate in such circumstances and this shortcoming has prompted the search for a more flexible labeling in classification problems. The theory of fuzzy sets was suggested as a way to remedy this difficulty. The seminal publication reporting the use of fuzzy sets in pattern recognition was that of Bellman *et al.* [4]. Since then the combination of fuzzy sets and pattern classification has been studied by many researchers [5], [6], [26]. The flexibility of fuzzy sets and the computational efficiency of neural networks with their proven record in pattern recognition problems has caused a great amount of interest in the combination of the two [2], [6], [9], [10], [17], [22], [25], [27], [30], [31].

The pattern recognition method reported in this paper originated from our investigation into uncertain information processing in the context of decision support (DS) for operational control of industrial processes [4], [12]. The essential requirements of such a system were: the ability to process input data in form of confidence limits; the ability to incorporate new information without need to completely retrain the network; and the ability to combine the supervised and unsupervised learning strategies within a single algorithm.

The fuzzy min-max (FMM) clustering and classification neural networks [30], [31], with their representation of classes as hyperboxes in $n$-dimensional pattern space and their conceptually simple but powerful learning process, provided a natural basis for our development. Interesting derivatives of the original FMM can also be found in [17] and [21].

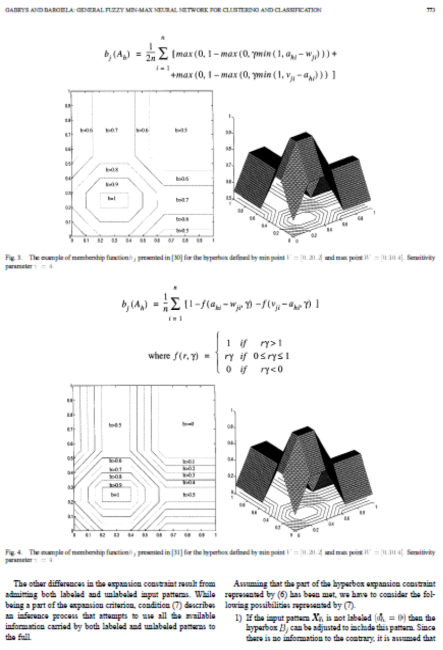
The proposed generalized fuzzy min-max (GFMM) neural network incorporates significant modifications that improve the effectiveness of the original algorithms. In developing the

Manuscript received February 2, 1998; revised January 15, 1999 and June 22, 1999.

The authors are with the Real Time Telemetry Systems, Department of Computing, Nottingham Trent University, Nottingham NG1 4BU, U.K.

Publisher Item Identifier S 1045-9227(00)04929-8.

1045–9227/00$10.00 © 2000 IEEE



GABRYS AND BARGIELA: GENERAL FUZZY MIN-MAX NEURAL NETWORK FOR CLUSTERING AND CLASSIFICATION 773

$$b_j(A_h) = \frac{1}{2n}\sum_{i=1}^{n}[max(0, 1-max(0, \gamma min(1, a_{hi}-w_{ji})))+$$
$$+max(0, 1-max(0, \gamma min(1, v_{ji}-a_{hi})))]$$

Fig. 3. The example of membership function $b_j$ presented in [30] for the hyperbox defined by min point $V = [0.2\ 0.2]$ and max point $W = [0.3\ 0.4]$. Sensitivity parameter $\gamma = 4$.

$$b_j(A_h) = \frac{1}{n}\sum_{i=1}^{n}[1-f(a_{hi}-w_{ji}, \gamma)-f(v_{ji}-a_{hi}, \gamma)]$$

$$\text{where } f(r,\gamma) = \begin{cases} 1 & if \quad r\gamma>1 \\ r\gamma & if \quad 0\le r\gamma\le 1 \\ 0 & if \quad r\gamma<0 \end{cases}$$

Fig. 4. The example of membership function $b_j$ presented in [31] for the hyperbox defined by min point $V = [0.2\ 0.2]$ and max point $W = [0.3\ 0.4]$. Sensitivity parameter $\gamma = 4$.

The other differences in the expansion constraint result from admitting both labeled and unlabeled input patterns. While being a part of the expansion criterion, condition (7) describes an inference process that attempts to use all the available information carried by both labeled and unlabeled patterns to the full.

Assuming that the part of the hyperbox expansion constraint represented by (6) has been met, we have to consider the following possibilities represented by (7).

1) If the input pattern $X_h$ is not labeled ($d_h = 0$) then the hyperbox $B_j$ can be adjusted to include this pattern. Since there is no information to the contrary, it is assumed that

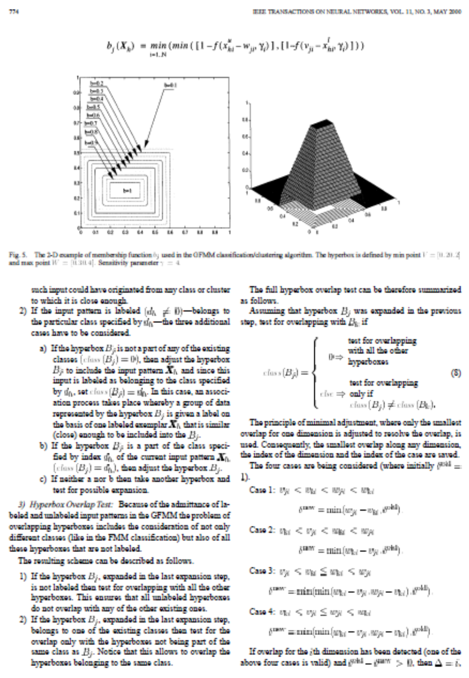
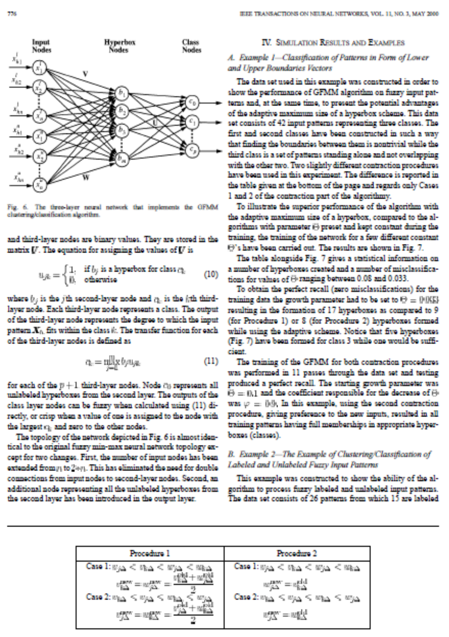
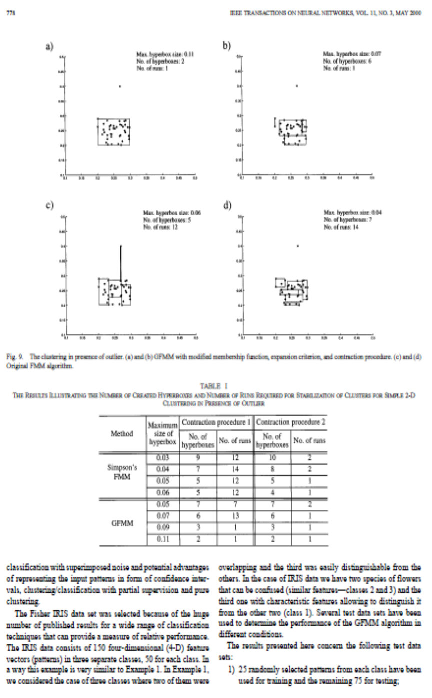




*Figure 9 – Some remarkable pages of the paper [18]*

For the research process, the images of the text document are not going to be taken from the screen of any device. All the text documents used are previously converted to a grey scale image in order to simplify the programming involved. Then, they are converted to the negative image. The reason for this last conversion is related with the greyscale colours 0 is pure black and 255 is pure white, thus, an area where pixels are greater than 0 has information. It simplifies the computation involved in the algorithms.

## 5.2 Finding the optimal image resolution to work with. Downsampling.

The images can be converted from the documents to any resolution, but in order to work faster and lighter (in terms of memory) it is necessary to establish an effective downsampling rate. When a text document is converted to an image it is actually a matrix in which all of its elements represent a coloured pixel. A higher resolution will result in an excellent detection of the areas of information but will force the computation to handle huge matrixes. It will produce very slow processing times or memory saturation. Instead, a low resolution will be faster and lighter for the memory, but the detection of the areas of information can be too poor and make the application fail.

The document 1 was taken as a reference example due to the diversity of information contained and the fact that it has a two columns format with the narrowest space between them and the narrowest interline space. It was the most demanding document (in the related terms) so if the methodology works well with it, it would work for the rest of the examples. Some experiments were done with different resolutions.

The resolution of an image can be set in several ways. You can fix a number of pixels in the horizontal or vertical dimensions, but there is a risk to deform the image. The simplest way of setting a resolution (and probably the best) is through ppi rate (pixels per inch) with the restriction of keeping the proportion of the image.

I tried 600, 70 and 50 ppi images with the option of smoothing the edges on and off.

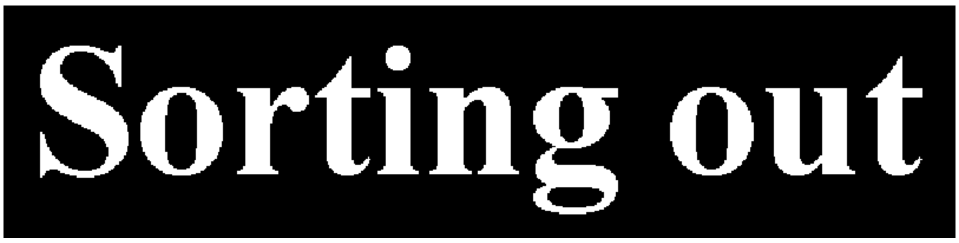


*Figure 10 – Original image*

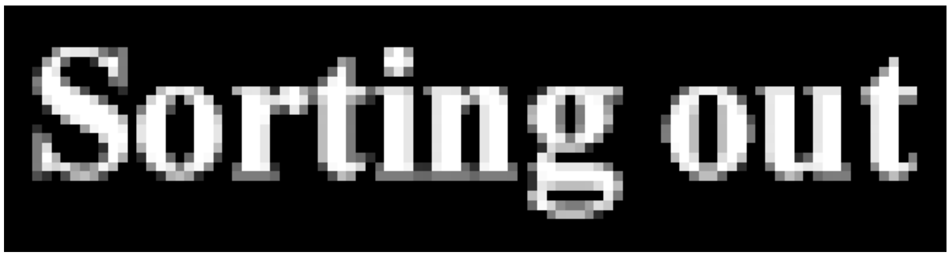


*Figure 11 – Converted to 70 ppi with smoothing option*

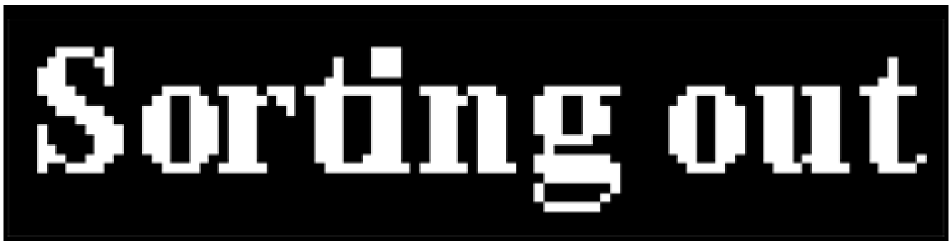


*Figure 12 – Converted to 70 ppi without smoothing option*

Before continuing with election of the optimal resolution, I will explain properly the two downsampling options and its results.

### *5.2.1 Smoothing edges in downsampling the image to lower resolutions*

The smoothing effect related with images can be achieved in several ways. Here, it was used the one described by R. Keys in [29] about the bicubic convolution interpolation applied to image processing.

In mathematics, bicubic interpolation is an extension of cubic interpolation for interpolating data points on a two dimensional regular grid. The interpolated surface is smoother than corresponding surfaces obtained by bilinear interpolation or nearest-neighbour interpolation. Bicubic interpolation can be accomplished using either Lagrange polynomials, cubic splines or cubic convolution algorithm.

In image processing, bicubic interpolation is often chosen over bilinear interpolation or nearest neighbour in image resampling, when speed is not an issue. Images resampled with bicubic interpolation are smoother and have fewer interpolation artifacts.

Suppose the function values f and the derivatives *fx, fy* and *fxy* are known at the four corners (0,0), (0,1), (1,0) and (1,1) of the unit square. The interpolated surface can then be written

$$p(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij} x^i y^j.$$

The interpolation problem consists in determining the 16 coefficients $a_{ij}$. Matching *p(x,y)* with the function values yields four equations,

$$\begin{aligned} f(0,0) &= p(0,0) = a_{00} \\ f(1,0) &= p(1,0) = a_{00} + a_{10} + a_{20} + a_{30} \\ f(0,1) &= p(0,1) = a_{00} + a_{01} + a_{02} + a_{03} \\ f(1,1) &= p(1,1) = \textstyle\sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij} \end{aligned}$$

Likewise, eight equations for the derivatives in the *x*-direction and the *y*-direction

$$\begin{aligned}
f_x(0,0) &= p_x(0,0) = a_{10} \\
f_x(1,0) &= p_x(1,0) = a_{10} + 2a_{20} + 3a_{30} \\
f_x(0,1) &= p_x(0,1) = a_{10} + a_{11} + a_{12} + a_{13} \\
f_x(1,1) &= p_x(1,1) = \textstyle\sum_{i=1}^{3}\sum_{j=0}^{3} a_{ij} i \\
f_y(0,0) &= p_y(0,0) = a_{01} \\
f_y(1,0) &= p_y(1,0) = a_{01} + a_{11} + a_{21} + a_{31} \\
f_y(0,1) &= p_y(0,1) = a_{01} + 2a_{02} + 3a_{03} \\
f_y(1,1) &= p_y(1,1) = \textstyle\sum_{i=0}^{3}\sum_{j=1}^{3} a_{ij} j
\end{aligned}$$

and four equations for the cross derivative *xy*.

$$\begin{aligned}
f_{xy}(0,0) &= p_{xy}(0,0) = a_{11} \\
f_{xy}(1,0) &= p_{xy}(1,0) = a_{11} + 2a_{21} + 3a_{31} \\
f_{xy}(0,1) &= p_{xy}(0,1) = a_{11} + 2a_{12} + 3a_{13} \\
f_{xy}(1,1) &= p_{xy}(1,1) = \textstyle\sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij} ij
\end{aligned}$$

where the expressions above have used the following identities,

$$\begin{aligned}
p_x(x,y) &= \textstyle\sum_{i=1}^{3}\sum_{j=0}^{3} a_{ij} i x^{i-1} y^j \\
p_y(x,y) &= \textstyle\sum_{i=0}^{3}\sum_{j=1}^{3} a_{ij} x^i j y^{j-1} \\
p_{xy}(x,y) &= \textstyle\sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij} i x^{i-1} j y^{j-1}
\end{aligned}$$

This procedure yields a surface *p(x,y)* on the unit square [0,1] x [0,1] which is continuous and with continuous derivatives. Bicubic interpolation on an arbitrarily sized regular grid can then be accomplished by patching together such bicubic surfaces, ensuring that the derivatives match on the boundaries.

If the derivatives are unknown, they are typically approximated from the function values at points neighbouring the corners of the unit square, e.g. using finite differences.

Grouping the unknown parameters $a_{ij}$ in a vector,

$$\alpha = \begin{bmatrix} a00 & a10 & a20 & a30 & a01 & a11 & a21 & a31 & a02 & a12 & a22 & a32 & a03 & a13 & a23 & a33 \end{bmatrix}^T$$

and letting

$$x = \begin{bmatrix} f(0,0) & f(1,0) & f(0,1) & f(1,1) & f_x(0,0) & f_x(1,0) & f_x(0,1) & f_x(1,1) & f_y(0,0) & f_y(1,0) & f_y(0,1) & f_y(1,1) & f_{xy}(0,0) & f_{xy}(1,0) & f_{xy}(0,1) & f_{xy}(1,1) \end{bmatrix}^T$$

the problem can be reformulated into a linear equation *Aα=x* where its inverse is:

$$A^{-1} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-3 & 3 & 0 & 0 & -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & -2 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 3 & 0 & 0 & -2 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 & 1 & 1 & 0 & 0 \\
-3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -1 & 0 \\
9 & -9 & -9 & 9 & 6 & 3 & -6 & -3 & 6 & -6 & 3 & -3 & 4 & 2 & 2 & 1 \\
-6 & 6 & 6 & -6 & -3 & -3 & 3 & 3 & -4 & 4 & -2 & 2 & -2 & -2 & -1 & -1 \\
2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
-6 & 6 & 6 & -6 & -4 & -2 & 4 & 2 & -3 & 3 & -3 & 3 & -2 & -1 & -2 & -1 \\
4 & -4 & -4 & 4 & 2 & 2 & -2 & -2 & 2 & -2 & 2 & -2 & 1 & 1 & 1 & 1
\end{bmatrix}$$

Bicubic spline interpolation requires the solution of the linear system described above for each grid cell. An interpolator with similar properties can be obtained by applying a convolution with the following kernel in both dimensions:

$$W(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & \text{for } |x| \leq 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & \text{for } 1 < |x| < 2 \\ 0 & \text{otherwise} \end{cases}$$

where α is usually set to -0.5 or -0.75. Note that $\omega(0) = 1$ and $\omega(n) = 0$ for all nonzero integers *n*.

If we use the matrix notation for the common case $\alpha = -0.5$ , we can express the equation using matrix notation:

$$p(t) = \tfrac{1}{2}\begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} a_{-1} \\ a_0 \\ a_1 \\ a_2 \end{bmatrix}$$

$t$ between 0 and 1 for one dimension. For two dimensions it is applied first in *x* and then in *y:*

$$\begin{aligned} b_{-1} &= p(t_x, a_{(-1,-1)}, a_{(0,-1)}, a_{(1,-1)}, a_{(2,-1)}) \\ b_0 &= p(t_x, a_{(-1,0)}, a_{(0,0)}, a_{(1,0)}, a_{(2,0)}) \\ b_1 &= p(t_x, a_{(-1,1)}, a_{(0,1)}, a_{(1,1)}, a_{(2,1)}) \\ b_2 &= p(t_x, a_{(-1,2)}, a_{(0,2)}, a_{(1,2)}, a_{(2,2)}) \\ p(x,y) &= p(t_y, b_{-1}, b_0, b_1, b_2) \end{aligned}$$

### *5.2.2 Sharpening edges in downsampling the image to lower resolutions*

Nearest-neighbour interpolation (also known as proximal interpolation) is a simple method of multivariate interpolation in one or more dimensions. Interpolation is the problem of approximating the value for a non-given point when the grid of pixels is reduced (or augmented).

It is fast and very simple to implement. The information it needs are both the horizontal and vertical ratios between the original image and the (to be) scaled image. If $w_1$ and $h_1$ are the width and height of the original image, whereas $w_2$ and $h_2$ are the width and height when enlarged (or shrinked). Calculating the ratio for both horizontal and vertical plane is given by,

$$\begin{cases} ratio_x = \dfrac{w_1}{w_2} \\ ratio_y = \dfrac{h_1}{h_2} \end{cases} \quad w_2, h_2 \neq 0$$

Once ratio has been calculated, the nearest neighbour algorithm selects the value of the nearest point and does not consider the values of neighbouring points at all.

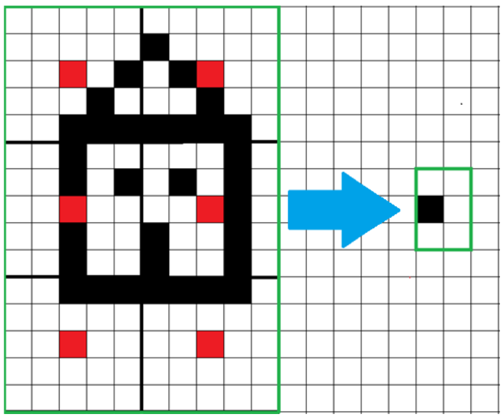


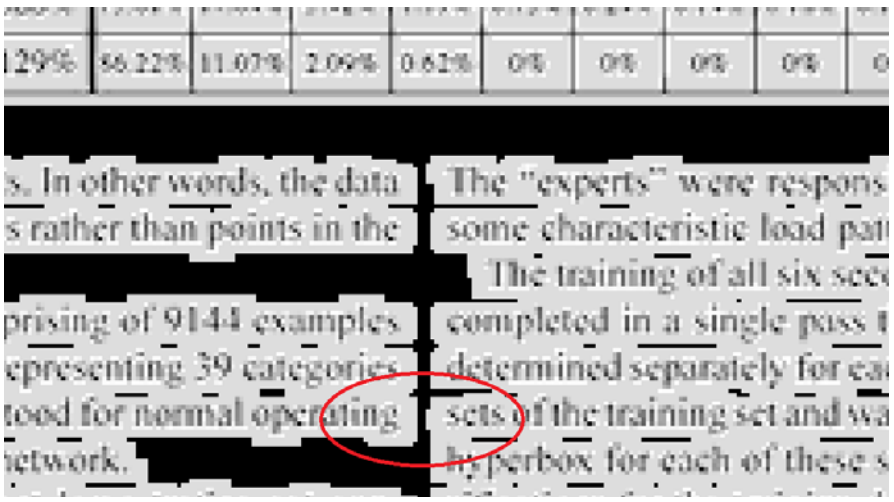
*Figure 13 – Graphical example of the proximal interpolation*

For more detailed information the reader is addressed to look up the research by Rukundo Oliver in [30].

Returning to optimal resolution election, a resolution of 600 ppi was easily rejected because a document of 15 pages supposed a matrix of 99000 x 5100 which means very high computational costs.

The comparison will be between 70 ppi smoothed, 70 ppi not smoothed, 50 ppi smoothed and 50 ppi not smoothed. In the following figures we can see an example of the results and why 70 ppi smoothed is the final election.

The remarked detail in the red ellipses explains the preference for the 70 ppi, which keeps the elements perfectly isolated. The gaps between columns are 3 pixels which are reliable in case some of the boundary could stand out.



| 129% | 86.22% | 11.07% | 2.09% | 0.62% | 0% | 0% | 0% | 0% | 0 |
|---|---|---|---|---|---|---|---|---|---|

s. In other words, the data
s rather than points in the
prising of 9144 examples
epresenting 39 categories
tood for normal operating
etwork.

The "experts" were respons
some characteristic load pat
The training of all six sec
completed in a single pass t
determined separately for ea
sets of the training set and wa
hyperbox for each of these s

*Figure 14 –Smoothed 70 ppi*

With 50 ppi resolution there is a risk of merging the columns. Notice that the space between them is only 1 pixel.

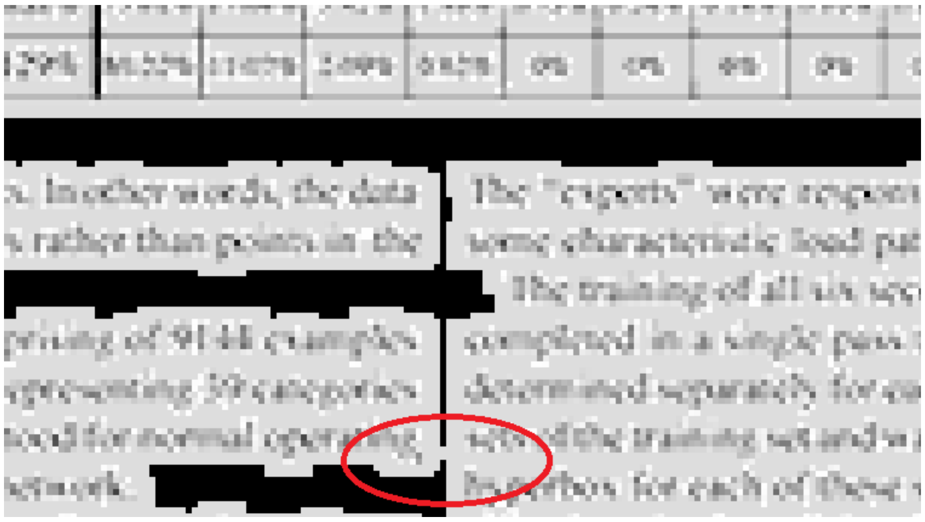


*Figure 15 – Smoothed 50 ppi*

Using the not smoothing option makes disappear some important traces of the fonts and lines, which subsequently will carry fails in the methodology.

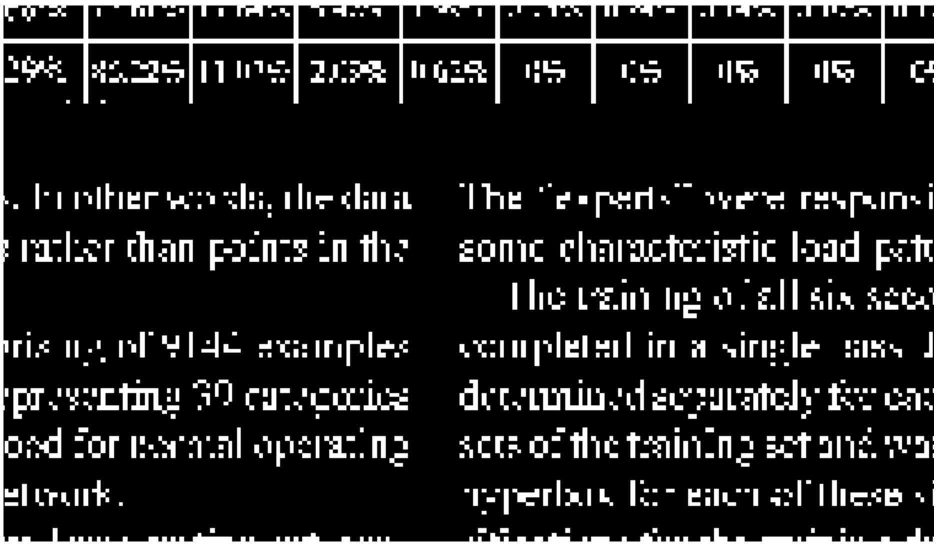


*Figure 16 – Not Smoothed 70 ppi*

## 5.3 Extracting the areas of information from the background. Dilation

It is essencial that the areas where the text, formulas, graphics, images, tables… were recognized and kept divided correctly in order to achieve an appropriate posterior sorting.

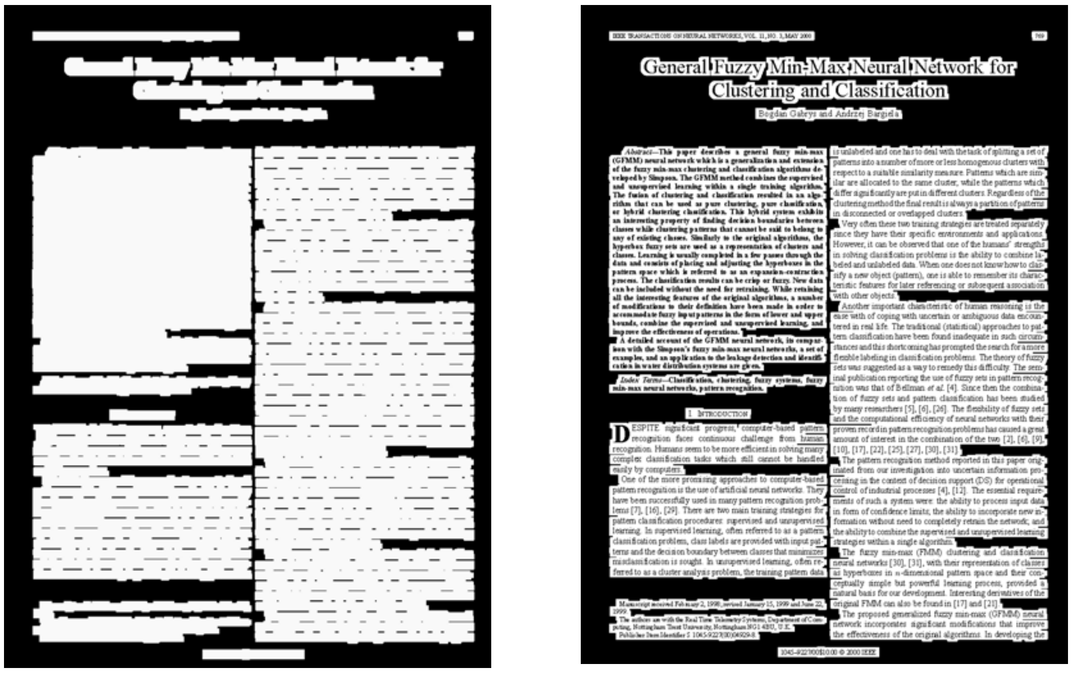

General Fuzzy Min-Max Neural Network for Clustering and Classification
Bogdan Gabrys and Andrzej Bargiela

*Figure 17 – Areas with information and the same areas with the corresponding information overlayed*

The detection of those areas is possible thanks to the effect of a mathematical morphology named dilation. Intuitively it is an area with fixed dimensions that will go through every possible position in the document and will differentiate the areas where it can be placed without having intersection with the text, graphics… and where it would have intersection.

### 5.3.1 Dilation [31]

Dilation is a morphological operation used to enhance the features of an image. Its function requires two inputs, an image to be dilated and a two dimensional structuring element. The basic effect of the operator on a binary image is to gradually enlarge the boundaries of regions of foreground pixels.

The dilation second input, the structuring element, is also known as dilation kernel. It is the element that determines the precise effect of the dilation on the input image.

In binary morphology, dilation is a shift-invariant (translation invariant) operator, strongly related to Minkowski's addition [32]. A binary image is viewed in mathematical morphology as a subset of an Euclidean space $R^d$ or the integer grid $Z^d$, for some dimension $d$.

Let $E$ be a Euclidean space or an integer grid, $A$ a binary image in $E$, and $B$ a structuring element.

The dilation of $A$ by $B$ is defined by:

$$A \oplus B = \bigcup_{b \in B} A_b$$

If $B$ has a center on the origin, then the dilation of $A$ by $B$ can be understood as the emplacement of the points covered by $B$ when the center of $B$ moves inside $A$.

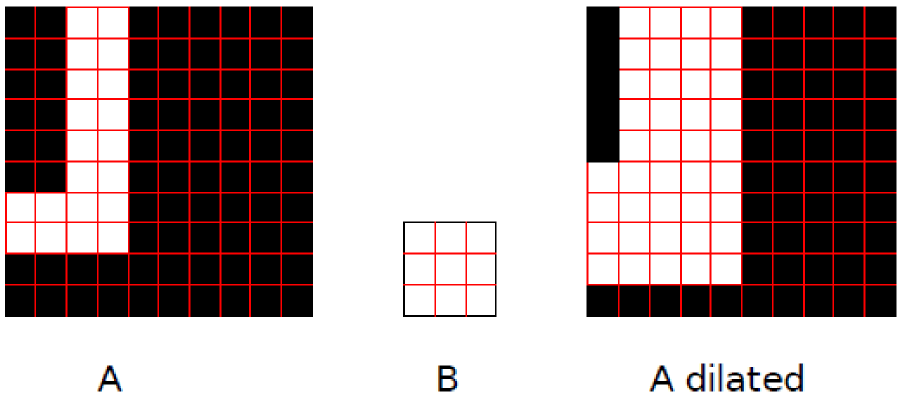


*Figure 18 – Original image A, kernel B and image A dilated*

The dilation can also be obtained by:

$$A \oplus B = \{z \in E | (B^s)_z \cap A \neq \varnothing\}$$

where $B^S$ denotes the symmetric of $B$, that is,

$$B^s = \{x \in E | -x \in B\}$$

As mentioned before, the structural element will determine the final effect of the dilation and it is basically a matrix of two dimensions. In order to get an appropriate dilation to our intent, every one of those dimensions can be related to two characteristic parameters of the document: standard interline space and the standard intercolumn space. Those parameters will be calculated from the document text through a specific methodology which involves statistical analysis described in the next point.

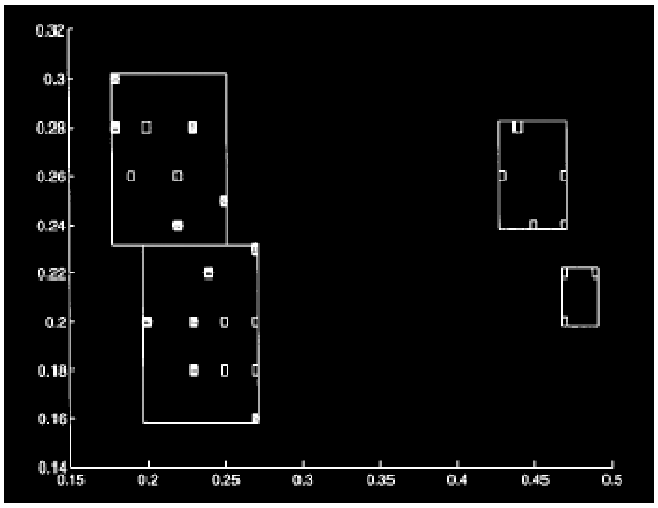


*Figure 19 – Graphic example of the original image as negative*

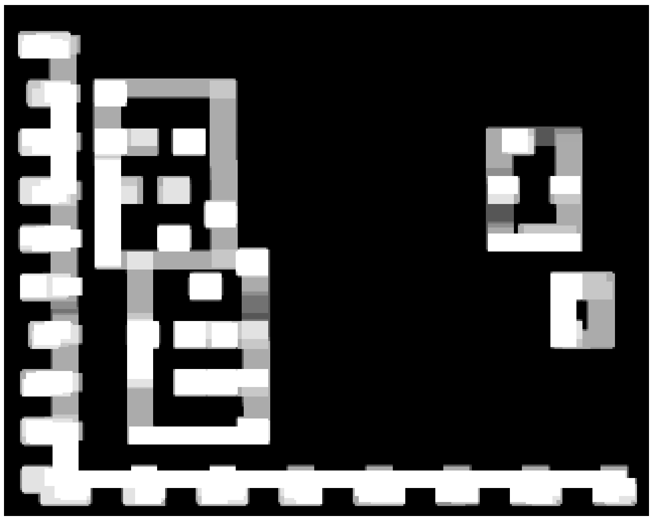


*Figure 20 – Graphic example of the dilation effect*

## 5.4 Extracting the standard interline space, the standard intercolumn space and the height of the standard line of the document. Dilation kernel.

The approach proposed is a personal approach although is based on statistical techniques. The idea of summing all the pixels of a binary image in the horizontal direction (or the projection in $y$) to get the interline space and summing all the pixels of a binary image in the vertical direction (or the projection in $x$) to get the intercolumn space is a solution that carries some setbacks that will be overcome later.

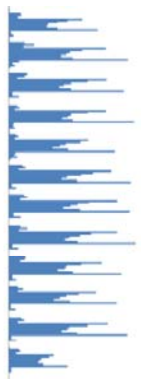
The previous conversion of the image to its negative takes sense in this point. Intuitively, we can think in white pixels as pixels where a printer would have to spend black ink (remember that the negative image is processed). The whiter the pixel is, the more ink that the printer would spend. Thus, if pure white is 255 and the pure black is 0, by summing the values of the pixels we can obtain histograms that will contain the information of interline and intercolumn spaces. The following step will be extracting that information.

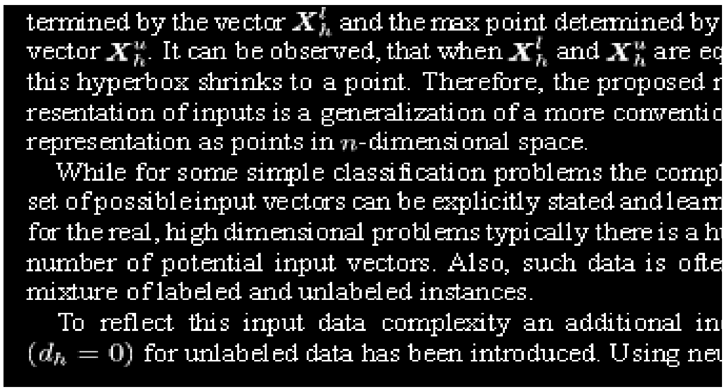
### *5.4.1 Extracting the standard interline space.*

Let $A_{hxw}$ be a matrix that represent the image, where h is the number of pixels vertically and w horizontally. Let $a_{hw} \in [0,255]$ be the elements of $A_{hxw}$ .

Let $S_{hor}$ be the sum of the pixels horizontally, a vector of one column and h rows.

$$S_{hor} = (h_1, \dots, h_h), \quad \text{with} \quad h_i = \sum_{i=1}^{w} a_{ij} \quad \text{and} \quad i = 1, \dots, h$$





termined by the vector $X_h^l$ and the max point determined by vector $X_h^u$. It can be observed, that when $X_h^l$ and $X_h^u$ are eq
this hyperbox shrinks to a point. Therefore, the proposed r
resentation of inputs is a generalization of a more conventio
representation as points in $n$-dimensional space.
While for some simple classification problems the compl
set of possible input vectors can be explicitly stated and learn
for the real, high dimensional problems typically there is a h
number of potential input vectors. Also, such data is ofte
mixture of labeled and unlabeled instances.
To reflect this input data complexity an additional in
$(d_h = 0)$ for unlabeled data has been introduced. Using ne

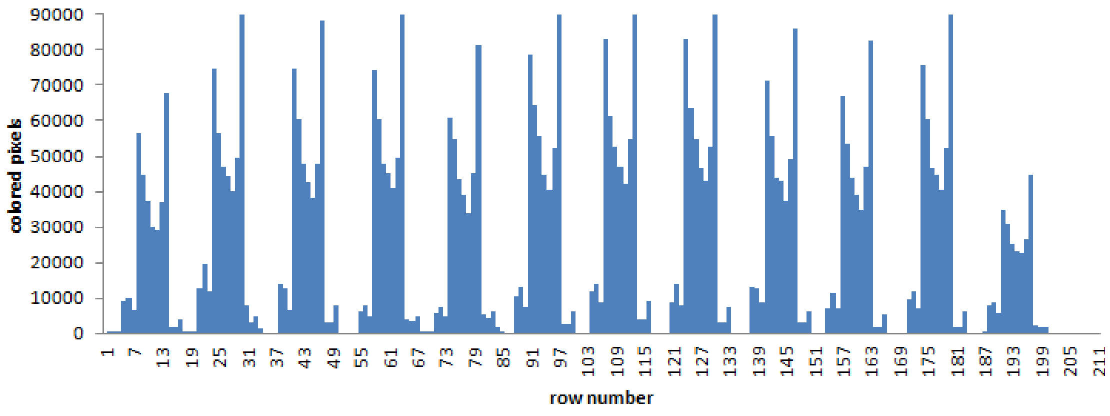


*Figure 21 – Histograms representing the sum of pixels horizontally*

In order to get the interline spaces $S_{hor}$ will be converted into a binary vector with the function

$$b_i = \begin{cases} 0, & \text{if } h_i < 0 \\ 1, & \text{if } h_i > 0 \end{cases}$$

$$bS_{hor} = (b_1, \dots, b_h),$$

Note this methodology can also be used with coloured background documents changing the threshold from to 0 to a parameter t.

Then, the vector $bS_{hor}$ is analyzed searching sequences like (…,0,1,…) related to the start of a line and (…,1,0,…) related to the end of a line. The function applied will be

$$if\ \left((b_i, b_{i+1}) = (0,1)\right) \rightarrow p_{ks} = i$$
$$if\ \left((b_i, b_{i+1}) = (1,0)\right) \rightarrow p_{ke} = i$$

being $\mathrm{p_s} = (\mathrm{p_1}, \dots, \mathrm{p_{ks}} \dots, \mathrm{p_n})$ a vector with the starting positions of the lines,
being $p_e = (p_1, \dots, p_{ke} \dots, p_n)$ a vector with the ending positions of the lines,

being n the number of lines.

The interline space is

$$ils = p_{s_{m+1}} - p_{e_m}\ with\ m{=}\ 1,\dots,\ n\text{-}1$$

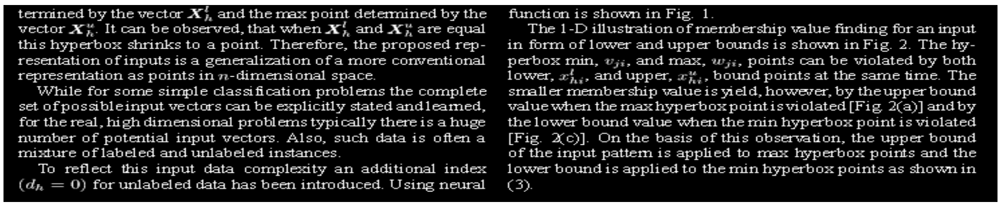
### 5.4.2 *Extracting the height of the standard line.*

The reasoning is the same used above but here the height is obtained

$$hl = p_{e_m} - p_{s_m}\ with\ m{=}\ 1,\dots,\ n$$

### 5.4.3 *Extracting the standard intercolumn space.*

It is possible to extract the intercolumn space applying the same fundamentals used for interline space. In this case, the projection is vertical so the formulas have to be adapted to columns instead of to rows.



termined by the vector $\boldsymbol{X}_h^l$ and the max point determined by the vector $\boldsymbol{X}_h^u$. It can be observed, that when $\boldsymbol{X}_h^l$ and $\boldsymbol{X}_h^u$ are equal this hyperbox shrinks to a point. Therefore, the proposed representation of inputs is a generalization of a more conventional representation as points in $n$-dimensional space.

While for some simple classification problems the complete set of possible input vectors can be explicitly stated and learned, for the real, high dimensional problems typically there is a huge number of potential input vectors. Also, such data is often a mixture of labeled and unlabeled instances.

To reflect this input data complexity an additional index $(d_h = 0)$ for unlabeled data has been introduced. Using neural function is shown in Fig. 1.

The 1-D illustration of membership value finding for an input in form of lower and upper bounds is shown in Fig. 2. The hyperbox min, $v_{ji}$, and max, $w_{ji}$, points can be violated by both lower, $x_{hi}^l$, and upper, $x_{hi}^u$, bound points at the same time. The smaller membership value is yield, however, by the upper bound value when the max hyperbox point is violated [Fig. 2(a)] and by the lower bound value when the min hyperbox point is violated [Fig. 2(c)]. On the basis of this observation, the upper bound of the input pattern is applied to max hyperbox points and the lower bound is applied to the min hyperbox points as shown in (3).

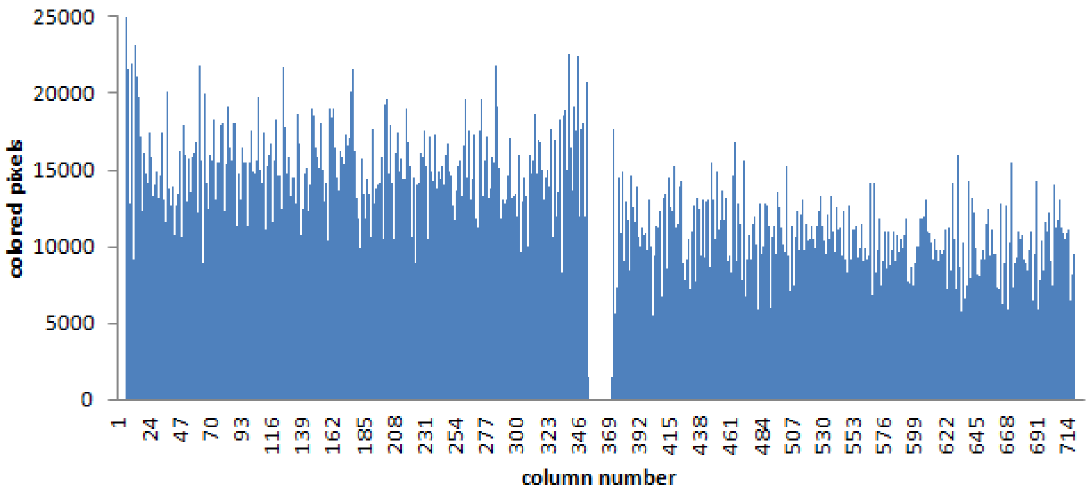
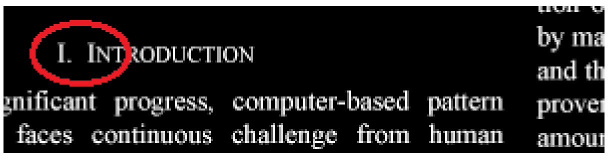


*Figure 23 – histograms representing the sum of pixels horizontally*

Extracting intercolumn space has a peculiarity with respect to the other values extraction. For example, the following figure shows a space that can be confused as intercolumn space (the space between ‘1.’ and ‘Introduction’).


I. INTRODUCTION

gnificant progress, computer-based pattern
faces continuous challenge from human

by ma
and th
prover
amour

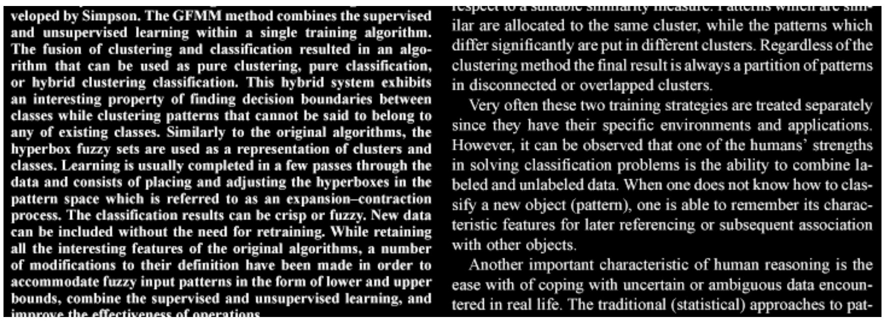
Figure 24 – Example of a mistaken intercolumn space

Then, it is necessary to add a condition to reject all the detected intercolumn spaces smaller to a determined parameter g. Intuitively it is set as to two times the standard line height Slh (in the following step it is explained how it is obtained Slh from lh). The experiments will confirm that g=2Slh is a good choice.

$$if\ (p_{s_{m+1}} - p_{e_m}) > g \ \rightarrow ilc = p_{s_{m+1}} - p_{e_m} \quad with \quad m = 1, \dots, n-1$$

### 5.4.4 Approach drawbacks

Some experiments show that this approach have its setbacks. It was easy the methodology to fail in most of the documents. The following images show the shows fail examples:


veloped by Simpson. The GFMM method combines the supervised and unsupervised learning within a single training algorithm. The fusion of clustering and classification resulted in an algorithm that can be used as pure clustering, pure classification, or hybrid clustering classification. This hybrid system exhibits an interesting property of finding decision boundaries between classes while clustering patterns that cannot be said to belong to any of existing classes. Similarly to the original algorithms, the hyperbox fuzzy sets are used as a representation of clusters and classes. Learning is usually completed in a few passes through the data and consists of placing and adjusting the hyperboxes in the pattern space which is referred to as an expansion–contraction process. The classification results can be crisp or fuzzy. New data can be included without the need for retraining. While retaining all the interesting features of the original algorithms, a number of modifications to their definition have been made in order to accommodate fuzzy input patterns in the form of lower and upper bounds, combine the supervised and unsupervised learning, and improve the effectiveness of operations.

respect to a suitable similarity measure. Patterns which are similar are allocated to the same cluster, while the patterns which differ significantly are put in different clusters. Regardless of the clustering method the final result is always a partition of patterns in disconnected or overlapped clusters.

Very often these two training strategies are treated separately since they have their specific environments and applications. However, it can be observed that one of the humans’ strengths in solving classification problems is the ability to combine labeled and unlabeled data. When one does not know how to classify a new object (pattern), one is able to remember its characteristic features for later referencing or subsequent association with other objects.

Another important characteristic of human reasoning is the ease with of coping with uncertain or ambiguous data encountered in real life. The traditional (statistical) approaches to pat-

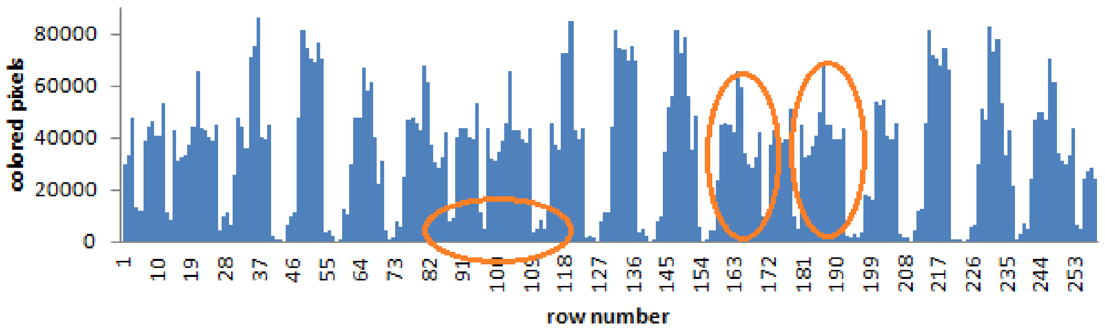


Figure 24 – The histogram represents the distortion in the interline space of the example

When the format of the page is two columns format, if one of the columns have different font formats, have an image, graphic or table; the white spaces between lines will be distorted. Some spaces will disappear and some line heights will be altered.

With the following example we can see that the methodology fails and it is not able to detect the intercolumn space.

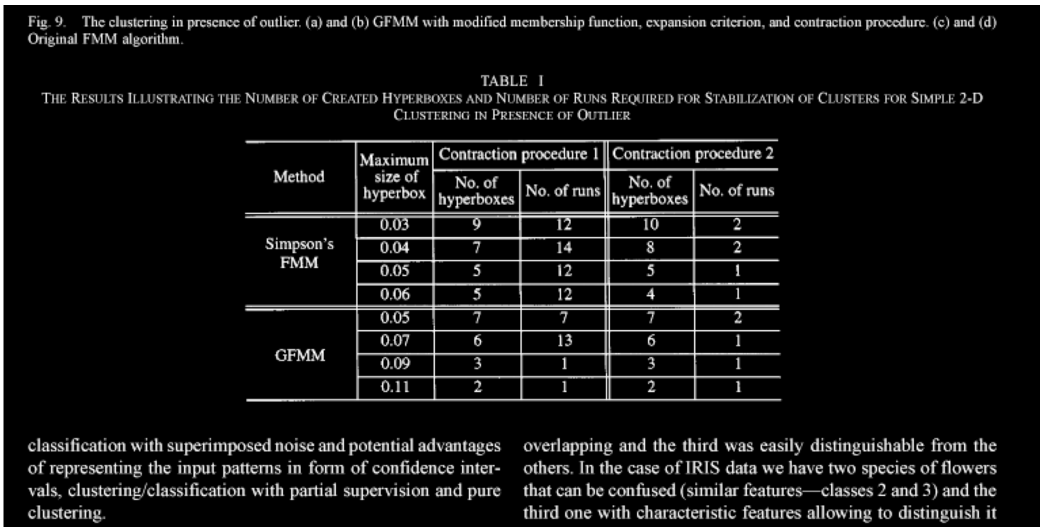


Fig. 9. The clustering in presence of outlier. (a) and (b) GFMM with modified membership function, expansion criterion, and contraction procedure. (c) and (d) Original FMM algorithm.

TABLE I
THE RESULTS ILLUSTRATING THE NUMBER OF CREATED HYPERBOXES AND NUMBER OF RUNS REQUIRED FOR STABILIZATION OF CLUSTERS FOR SIMPLE 2-D CLUSTERING IN PRESENCE OF OUTLIER

| Method | Maximum size of hyperbox | Contraction procedure 1 | | Contraction procedure 2 | |
|---|---|---|---|---|---|
| | | No. of hyperboxes | No. of runs | No. of hyperboxes | No. of runs |
| Simpson's FMM | 0.03 | 9 | 12 | 10 | 2 |
| | 0.04 | 7 | 14 | 8 | 2 |
| | 0.05 | 5 | 12 | 5 | 1 |
| | 0.06 | 5 | 12 | 4 | 1 |
| GFMM | 0.05 | 7 | 7 | 7 | 2 |
| | 0.07 | 6 | 13 | 6 | 1 |
| | 0.09 | 3 | 1 | 3 | 1 |
| | 0.11 | 2 | 1 | 2 | 1 |

classification with superimposed noise and potential advantages of representing the input patterns in form of confidence intervals, clustering/classification with partial supervision and pure clustering.

overlapping and the third was easily distinguishable from the others. In the case of IRIS data we have two species of flowers that can be confused (similar features—classes 2 and 3) and the third one with characteristic features allowing to distinguish it

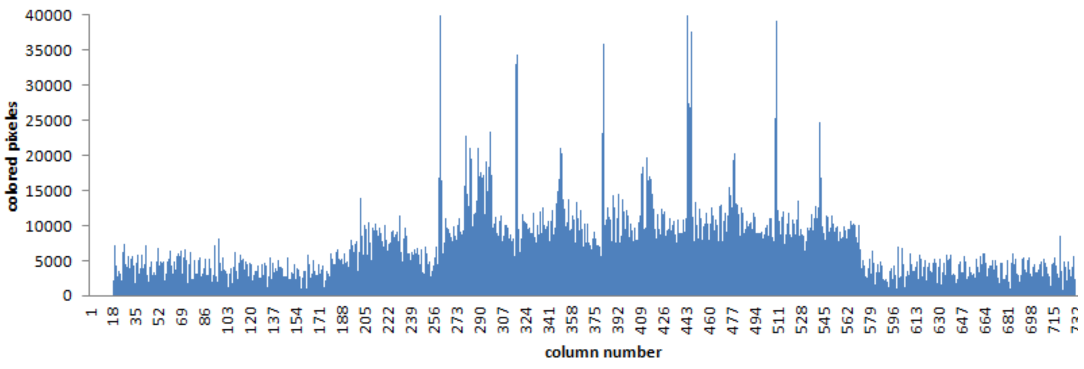


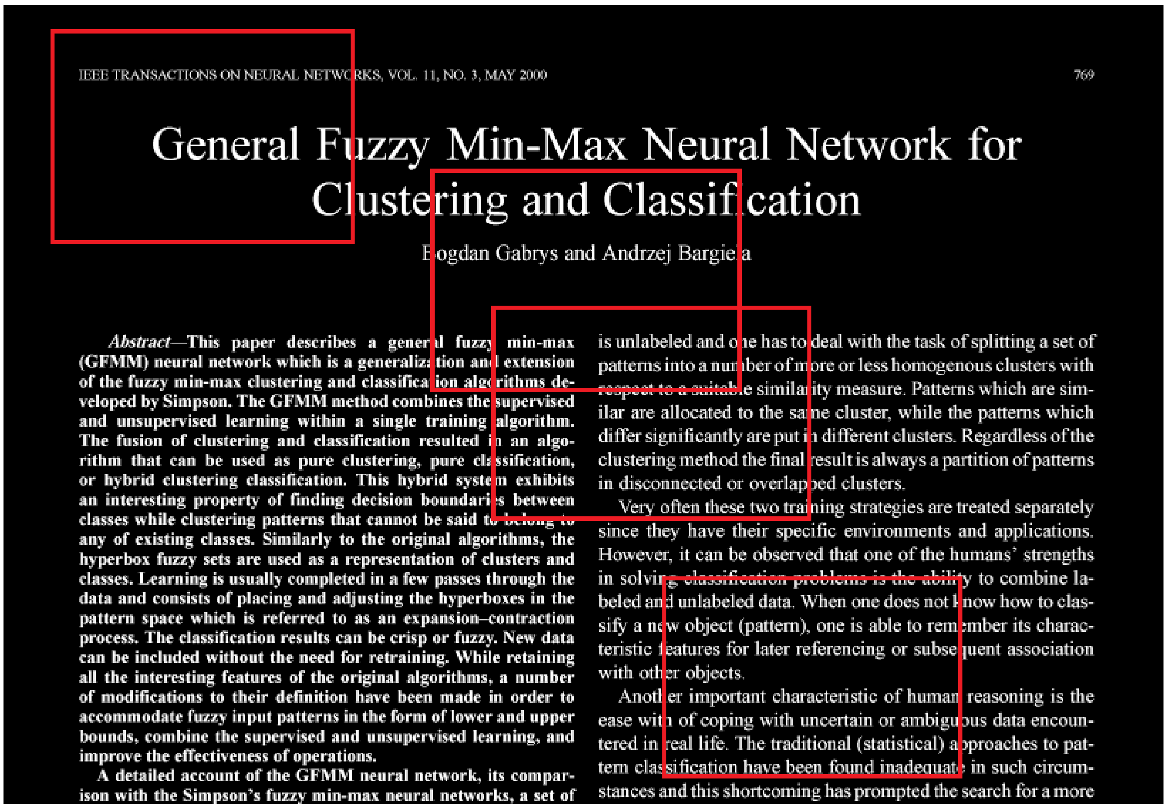
*Figure 25 – Example of a case where the intercolumn space is not detected*

## 5.4.5 *Working out the drawbacks*

The solution to avoid this kind of situations is inspired in the bootstrap concept but taking a final decision applying the statistical mode to the whole set of drawn samples.

Instead of analyzing the whole extension of the image, it will take smaller areas of it randomly (with resampling); and then, it would apply the previous ils, hl and ilc extraction technique to get the spaces.

It is important to control the bootstrap sampling somehow in order to have an effective methodology. If we consider w as the width and h as the height of one page of the document, the dimensions of the sample image is fixed to $\frac{1}{3}w$ in the horizontal dimension and $\frac{1}{5}h$ in the vertical dimension.



IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 11, NO. 3, MAY 2000 769

General Fuzzy Min-Max Neural Network for Clustering and Classification

Bogdan Gabrys and Andrzej Bargiela

*Abstract*—This paper describes a general fuzzy min-max (GFMM) neural network which is a generalization and extension of the fuzzy min-max clustering and classification algorithms developed by Simpson. The GFMM method combines the supervised and unsupervised learning within a single training algorithm. The fusion of clustering and classification resulted in an algorithm that can be used as pure clustering, pure classification, or hybrid clustering classification. This hybrid system exhibits an interesting property of finding decision boundaries between classes while clustering patterns that cannot be said to belong to any of existing classes. Similarly to the original algorithms, the hyperbox fuzzy sets are used as a representation of clusters and classes. Learning is usually completed in a few passes through the data and consists of placing and adjusting the hyperboxes in the pattern space which is referred to as an expansion–contraction process. The classification results can be crisp or fuzzy. New data can be included without the need for retraining. While retaining all the interesting features of the original algorithms, a number of modifications to their definition have been made in order to accommodate fuzzy input patterns in the form of lower and upper bounds, combine the supervised and unsupervised learning, and improve the effectiveness of operations.

A detailed account of the GFMM neural network, its comparison with the Simpson's fuzzy min-max neural networks, a set of

is unlabeled and one has to deal with the task of splitting a set of patterns into a number of more or less homogenous clusters with respect to a suitable similarity measure. Patterns which are similar are allocated to the same cluster, while the patterns which differ significantly are put in different clusters. Regardless of the clustering method the final result is always a partition of patterns in disconnected or overlapped clusters.

Very often these two training strategies are treated separately since they have their specific environments and applications. However, it can be observed that one of the humans' strengths in solving classification problems is the ability to combine labeled and unlabeled data. When one does not know how to classify a new object (pattern), one is able to remember its characteristic features for later referencing or subsequent association with other objects.

Another important characteristic of human reasoning is the ease with of coping with uncertain or ambiguous data encountered in real life. The traditional (statistical) approaches to pattern classification have been found inadequate in such circumstances and this shortcoming has prompted the search for a more

*Figure 26 – Representation of the different samples that could be taken randomly*

Obviously, some of the samples wouldn't return values for any of space, due to the previously commented drawbacks, so those samples would be rejected and the process would not end until there were stored n values for every *ils, ilc* and *hl.* (*n* was established as 500, so altogether 1500).

Let denote the mode of a sample space $\{q_1, \dots, q_{500}\}$ as $\tilde{q}$, so we would have $\widetilde{hl}$, $\widetilde{ils}$ and $\widetilde{ilc}$. Those three values need to be adjusted to work properly as the dilation kernel dimensions. Interline and intercolumn spaces are multiplied by 2 factors:

$$Standard\ interline\ space\ (Sils)\ =\ 2\,\widetilde{ils}$$

$$Standard\ intercolumn\ space\ (Sics) =\ 0.85\,\widetilde{ics}$$

Those correction factors are sensible because it is necessary to ensure that the dilation merges a text line and the one place below in a paragraph ($2\,\widetilde{ils}$) and it doesn't merge two columns ($0.85\,\widetilde{ics}$).

Finally, the dilation kernel can be established. If *K* is the matrix associated to the kernel, then *Sils* is the number of rows of *K* and *Sics* is the number of columns.

## 5.5 Binarizing the dilated image.

It is a very simple algorithm. If $A_{\mathrm{ij}}$ is the matrix of pixels corresponding to the original image and $a_{ij}$ the elements of $A_{ij}$, then $a_{\mathrm{ij}}$ represents the pixels of the image. Every pixel $b_{ij}$ of the matrix of pixels corresponding to the binarized image is calculated by the function

$$b_{ij} = \begin{cases} 0, & if\ a_{ij} < 0 \\ 255, & if\ a_{ij} > 0 \end{cases}$$

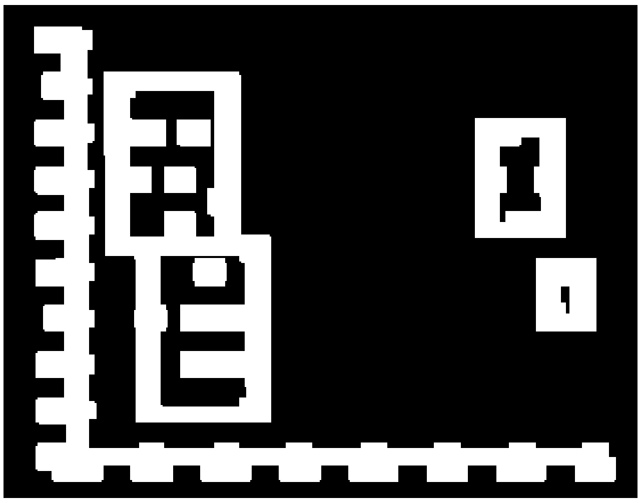


Figure 27 – Graphic example of binarization

## 5.6 Filling holes in the binarized image

There are several algorithms to do it, for example in [33],explained with graphic examples in the figure 28. It is necessary defining first what a hole is. A hole may be defined as a background region surrounded by a connected border of foreground pixels.

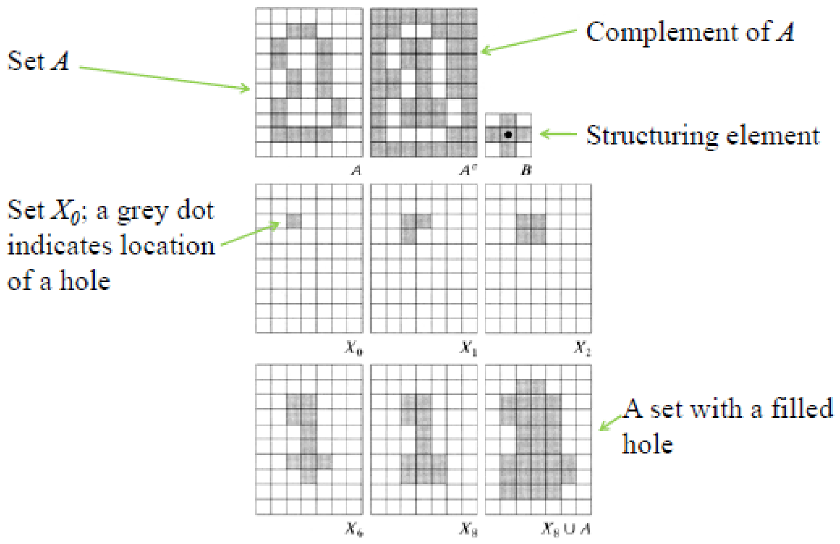


Figure 28 – Graphic example of the fill hole algorithm process

Let denote by $A$ a set whose elements are boundaries, each boundary encloses a background region (a hole). Given a point in each hole, the objective is to fill all the holes with ones (for binary images).

Starting from forming an array $X_0$ of zeros (the same size as the array containing $A$), except at the locations in $X_0$ corresponding to the given point in each hole, which is set to one. Then, the following procedure fills all the holes with ones:

$$X_k = \left(X_{k-1} \oplus B\right) \cap A^c \qquad k = 1, 2, 3, \ldots$$

in which $B$ is the structuring element or kernel. The working logic of the kernel is exactly the same that has in dilation but with a different purpose and final effect.

The algorithm terminates at the iteration step $k$ if $X_k = X_{k-1}$. The set $X_k$ then contains all the filled holes; the union of $X_k$ and A contains all the filled holes and their boundaries. The dilation would fill the entire area if left unchecked. However, the intersection at each step with the complement of $A$ limits the result to the inside of the region of interest.

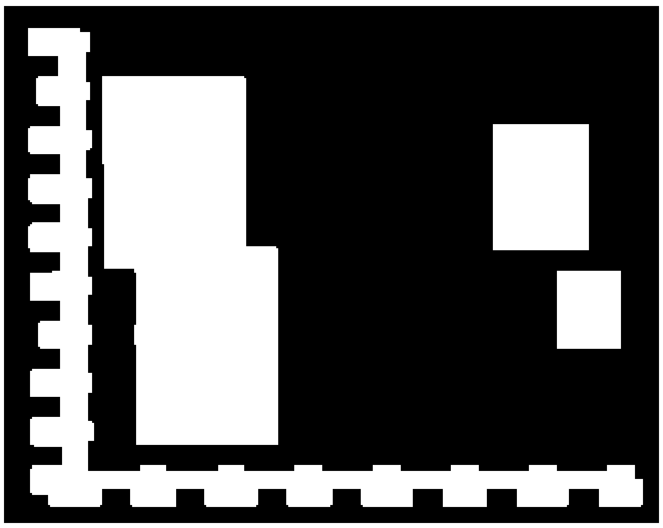


Figure 29 – Graphic example of a binary image with filled holes

## 5.7 Labeling not connected areas in the image with filled holes.

There are also several algorithms for labeling a binary image but it is going to be used the related in [34].

It iterates through 2-dimensional, binary data. The algorithm makes two passes over the image: one pass to record equivalences and assign temporary labels and the second to replace each temporary label by the label of its equivalence class.

Connectivity checks are carried out by checking the labels of pixels that are North-East, North, North-West and West of the current pixel (assuming 8-connectivity).

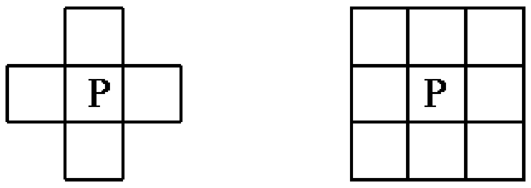


Figure 30 – Left: 4-conectivity. Right: 8-conectivity

4-connectivity uses only North and West neighbours of the current pixel. The following conditions are checked to determine the value of the label to be assigned to the current pixel (4-connectivity is assumed)

Conditions to check:

Does the pixel to the left (West) have the same value?

Yes - We are in the same region. Assign the same label to the current pixel

No - Check next condition

Do the pixels to the North and West of the current pixel have the same value but not the same label?

Yes - We know that the North and West pixels belong to the same region and must be merged. Assign the current pixel the minimum of the North and West labels, and record their equivalence relationship

No - Check next condition

Does the pixel to the left (West) have a different value and the one to the North the same value?

Yes - Assign the label of the North pixel to the current pixel

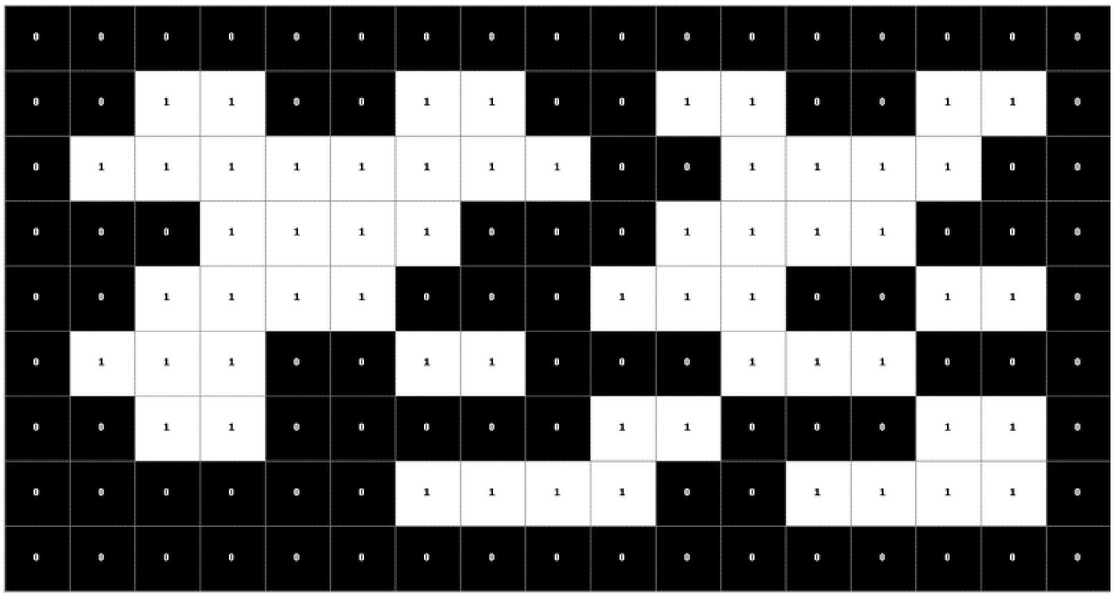
No - Check next condition

Do the pixel's North and West neighbours have different pixel values?

Yes - Create a new label id and assign it to the current pixel

This algorithm uses the union-find data structure which provides excellent performance for keeping track of equivalence relationships. Union-find essentially stores labels which correspond to the same module in a disjoint-set data structure, making it easy to remember the equivalence of two labels by the use of an interface method.

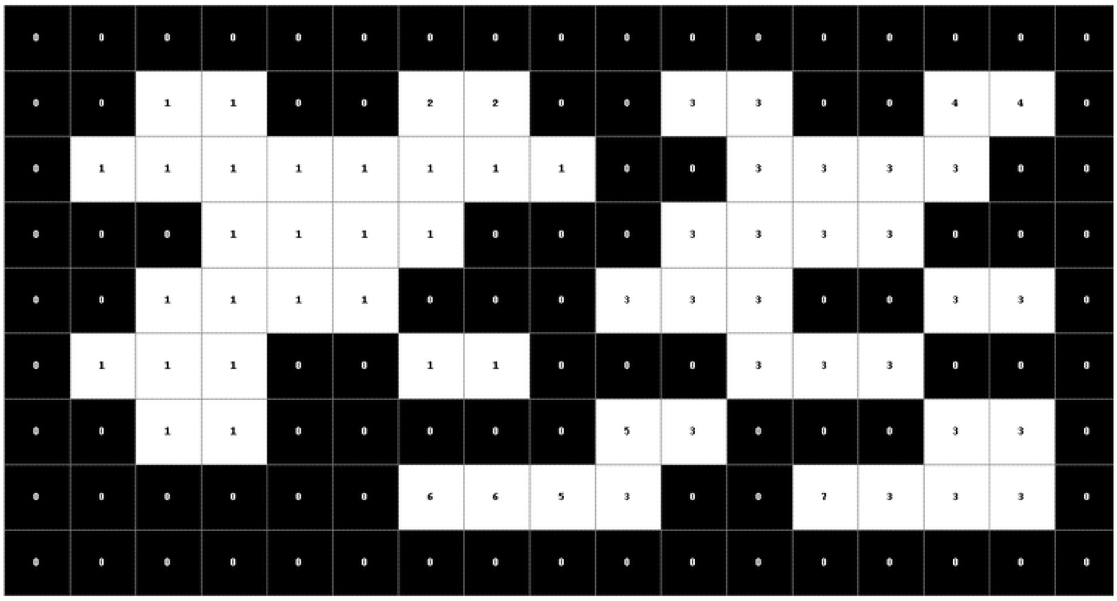
Once the initial labeling and equivalence recording is completed, the second pass merely replaces each pixel label with its equivalent disjoint-set representative element.



| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 31– Input binary image*

After the first pass, the following labels are generated. A total of 7 labels are generated in accordance with the conditions highlighted above



| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 2 | 2 | 0 | 0 | 3 | 3 | 0 | 0 | 4 | 4 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 3 | 3 | 3 | 3 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 3 | 3 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 5 | 3 | 0 | 0 | 0 | 3 | 3 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 6 | 6 | 5 | 3 | 0 | 0 | 7 | 3 | 3 | 3 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

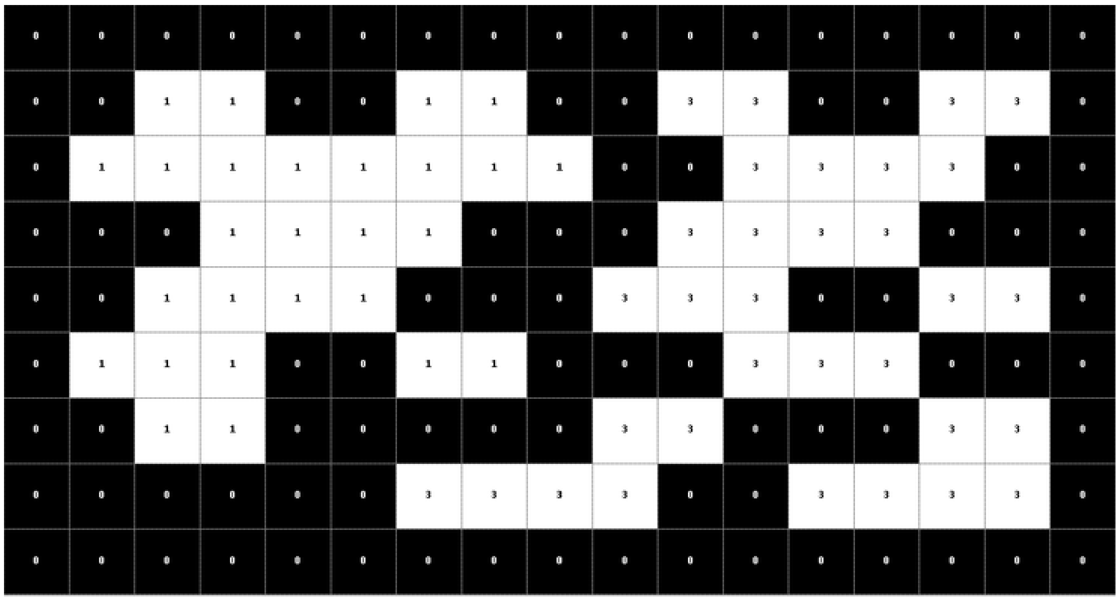
*Figure 32 – After the first pass of the algorithm*

The label equivalence relationships generated are,

| Set ID | Equivalent Labels |
|---|---|
| 1 | 1,2 |
| 2 | 1,2 |

| Set ID | Equivalent Labels |
|---|---|
| 3 | 3,4,5,6,7 |
| 4 | 3,4,5,6,7 |
| 5 | 3,4,5,6,7 |
| 6 | 3,4,5,6,7 |
| 7 | 3,4,5,6,7 |

Array generated after the merging of labels is carried out. Here, the label value that was the smallest for a given region "floods" throughout the connected region and gives two distinct labels, and hence two distinct labels.



| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 3 | 3 | 0 | 0 | 3 | 3 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 3 | 3 | 3 | 3 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 3 | 3 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 3 | 3 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 0 | 0 | 3 | 3 | 3 | 3 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 33 – After the second pass of the algorithm*

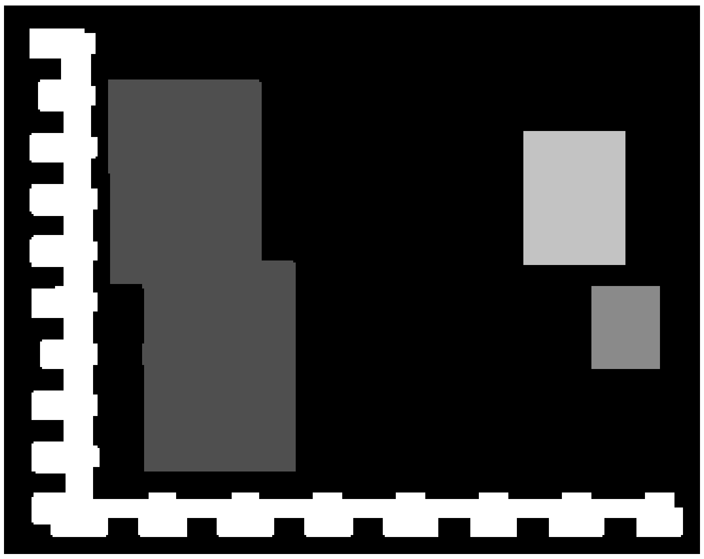


*Figure 34 – Graphic example of the labeling of a binary image with filled holes*

## 5.8 Joining related areas of information.

It is important for methodology to group some areas of interest, for example the areas with recognized elements inside. To get that task done:

Let B be the corresponding matrix to the binarized image and F the corresponding matrix to the labeled image with the holes filled.

Let be U the corresponding matrix to the labeled image with the holes unfilled. It is obtained with

$$U = B \cap F$$

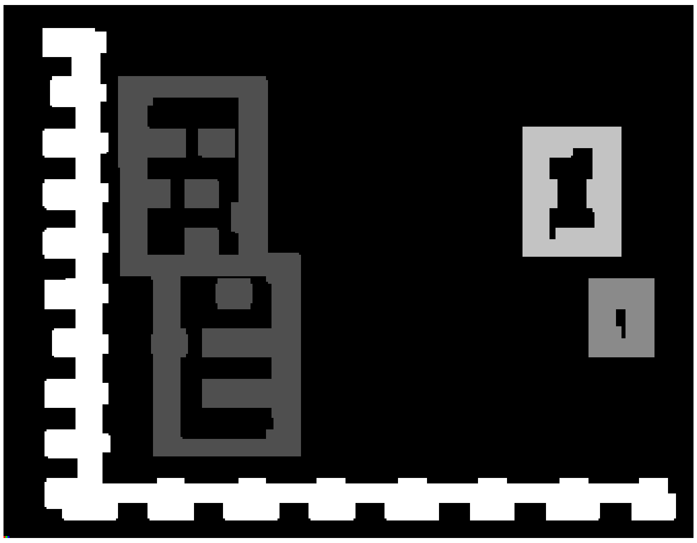


*Figure 35 – Graphic example of the labeling of a binary image with unfilled holes*

The image with filled holes and the image with unfilled holes has the same labels (notice the same colours in figure 34 and figure 35)

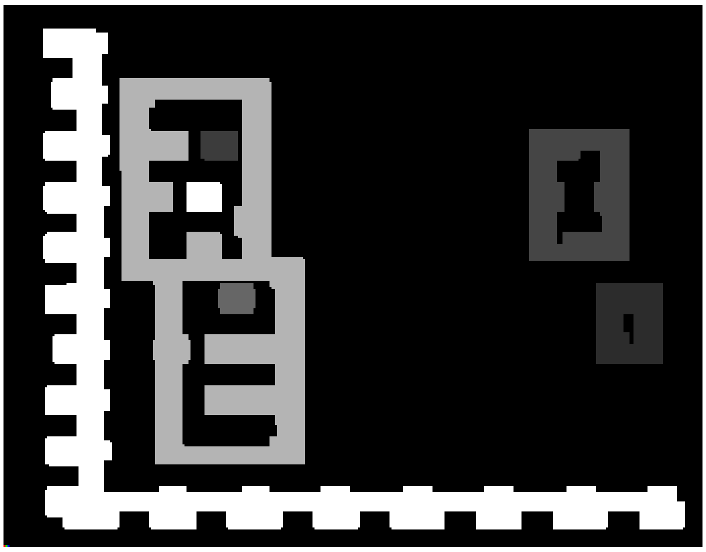


*Figure 1234 – Graphic example of labeling directly from the binary image with unfilled holes*

## 5.9 Sorting the areas of information

The sorting process of the areas of information detected is done in two steps. The first step sorts rough elements like columns, paragraphs, grahphics, tables, images, … and the second step is a finest sorting. It sorts the internal elements of every rough element sorted previously in the step one.

### *5.9.1 Sorting the rough elements of the information areas*

This is the main sorting step because a fail in this step would suppose a mess in the natural reading line which would carry a confusion in the user.

It is a personal approach and its fundaments are based in the analysis of the projection of the pixels in horizontal and in vertical as it was done in the extraction of the values for the intercolumn and interline space. Here the analisys is done incrementally instead of analysing the full image. The process would be executed in the binarized dilated image and can be understood easily with the following commands (find the command of every step in a list below):

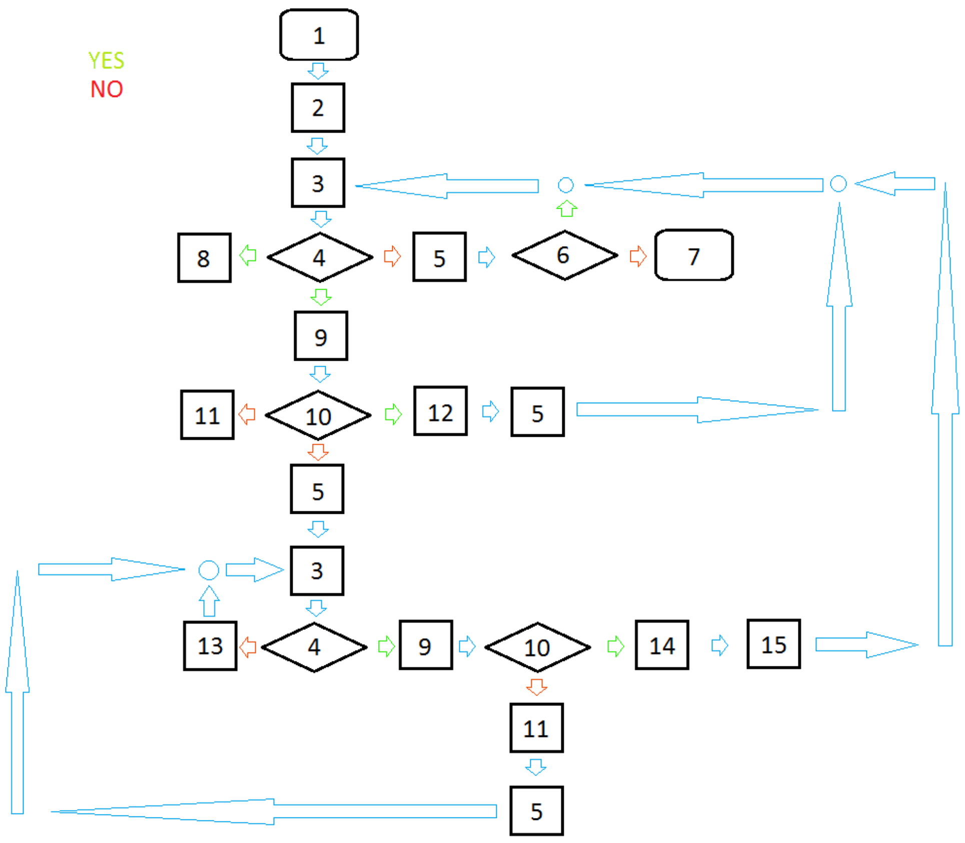


Figure 37 – Flow chart of the algorithm for rough sorting

| ***List of commands of the flow chart*** | |
|---|---|
| 1 | Start |
| 2 | i=0 |
| 3 | Analyse row i |
| 4 | Blank row? (it means the sum of all the elements of the row is equal to 0) |
| 5 | i++ |
| 6 | Is i equal the final row? |
| 7 | End |
| 8 | Save the position of the row as the begining of a rough element, b=i |
| 9 | Analyze the x proyection from b to i |
| 10 | Is there any parcial range in the proyection as 1,0,…,0,1? |
| 11 | Count the number of parcial ranges in the proyection as 1,0,…,0,1 and save their column position in the vector C |
| 12 | End of the rough element. The matrix which define the element E with rows from b to i and columns from 0 to the w (w is the total number of columns of the whole image) |
| 13 | e=i and i++ |
| 14 | End of the roughs element. Count the number of the n elements in C. There were n matrixes which define the n elements $E_n$ with rows from b to i and columns from $k_m$ to $k_{m+1}$ where k= $(0,C_1,\ldots,C_n,w)$ with m=1,…,n+1 |
| 15 | i=e |

The rough elements labels are stored in order in a vector ($\overline{rs}$).

### 5.9.2 Sorting the fine internal elements of each rough

For sorting the fine elements that are inside the rough detected areas, it is used a simple implementation that may cause some sorting errors comparing with the reading natural line.

The algorithm is very simple. It takes the elements with a specific label following the order of $\overline{rs}$. For every rough area of $\overline{rs}$, the inner elements are sorted from up to down. If different elements are in the shame vertical position, then the order is from left to right. The reference point of the elements for sorting is the upper left corner pixel. The labels are stored in the created order in a vector $(\overline{fs})$.

The percentage of fine sorting mistakes is almost inexistent and irrelevant in terms of making the reading a confusing process to the user.

For example in the next figure the internal elements of the graphic can be sorted wrongly but it would be irrelevant to the readability of the document.

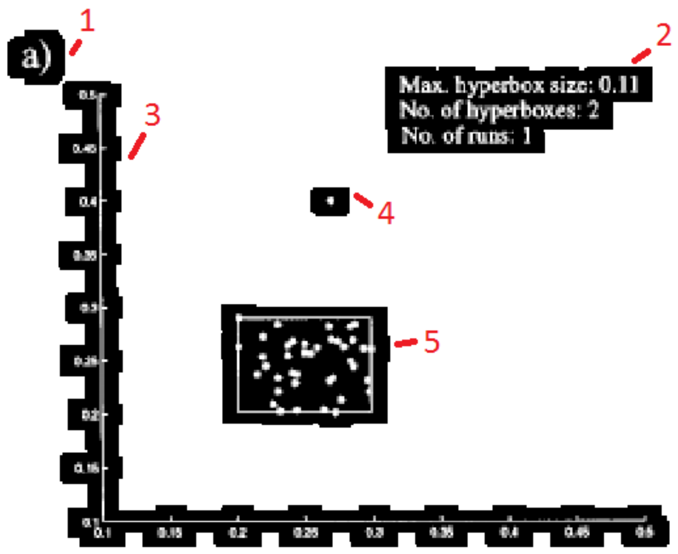
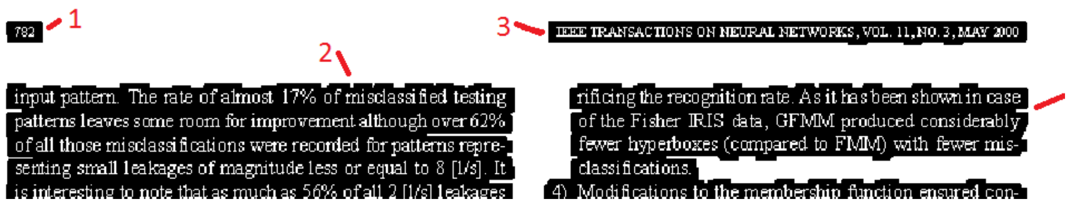


*Figure 38 – Graphic example of the minor confusion in a graphic*

Other typical example is the header of the pages when it has a two column format



782

IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 11, NO. 3, MAY 2000

input pattern. The rate of almost 17% of misclassified testing patterns leaves some room for improvement although over 62% of all those misclassifications were recorded for patterns representing small leakages of magnitude less or equal to 8 [l/s]. It is interesting to note that as much as 56% of all 2 [l/s] leakages

rificing the recognition rate. As it has been shown in case of the Fisher IRIS data, GFMM produced considerably fewer hyperboxes (compared to FMM) with fewer misclassifications.

4) Modifications to the membership function ensured con-

*Figure 39 – Graphic example of the typical confusion of the headers in some kind of page formats*

## 5.10 Determining the space of characteristics

Once every fine element of the rough areas of the document is sorted, the following step is to characterize them.

### *5.10.1 metrics*

The metrics used are:

the height of the detected area ($h$),

$$h = r_f - r_i$$

where $r_i$ = the minimum row of the pixels in the area $A$

and $r_f$ = the maximum row of the pixels in the area $A$

the density of the area considering the holes of the image unfilled $(d)$.

$$d = \frac{S}{A}$$

where $A$ is the area contained in the bounding box of a fine element of the rough areas and $S$ is

$$S = \sum_{i=r_i}^{r_f} \sum_{j=c_i}^{c_f} a_{ij}$$

with $c_i$ = the minimum column of the pixels in the area $A$

and $c_f$ = the maximum column of the pixels in the area $A$

### 5.10.2 Input space

The input space is a two dimensional space $X$ where $x_i \in X$ is

$$x_i = (h_i, d_i)$$

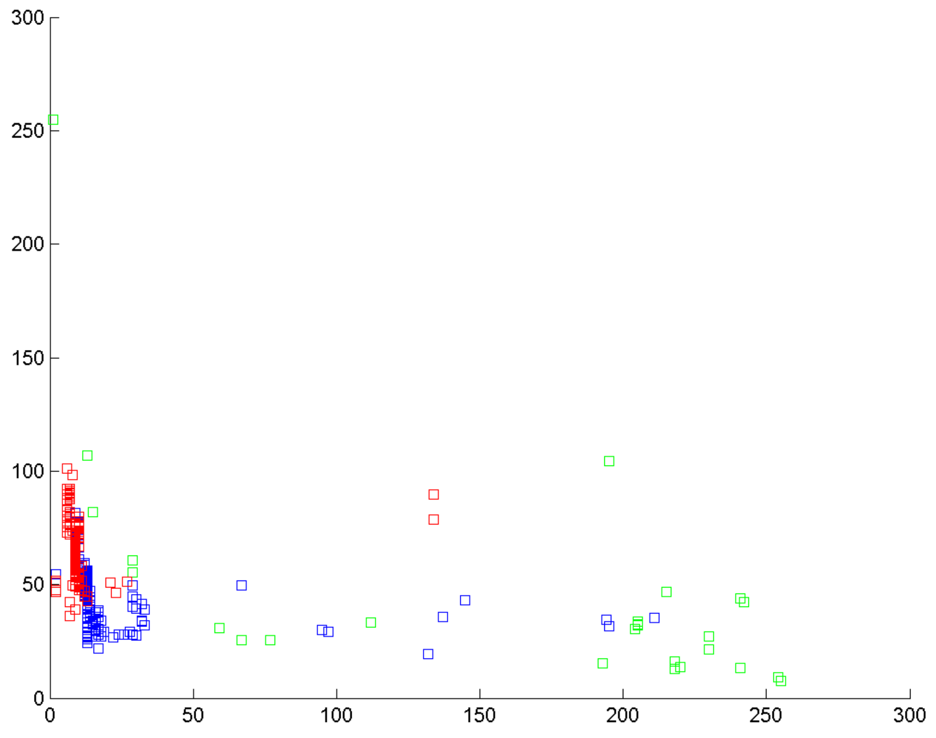


Figure 40 – Space of characteristics for dataset of the document 1

### *5.10.3 Output space*

First of all, it is necessary exercising some personal judgment about the classes in which the data want to be clustered. The number of classes is three and their fundaments are:

1. Major detail zooming (MaDZ) elements. Areas of information in which the user needs to zoom in to be able to read. This would necessary for line texts with small fonts like the typical headers, footers or captions. In the figures that illustrate the experiments and results they are represented as red.
2. Standard zooming (SZ) elements. Most of the document is readable with this specific zooming. For example, this zoom would correspond to the current line text. In the figures that illustrate the experiments and results they are represented as blue.
3. Minor detail zooming (MiDZ) elements. Areas of information in which the user needs to zoom out to be able read comfortably. For example, this zoom would suit for big images, graphics, titles, etc. In the figures that illustrate the experiments and results they are represented as green.

Thus, the output space is one dimensional space Y where $y_i \in Y$ is 1, 2 or 3 (the class explained above).

### *5.10.4 Space of characteristics*

Gathering the information about the input and output space, let *S* be the space of characteristics such that

$$S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

where the input space is two dimensional space *X*, $x_i \in X$ is $(h_i, d_i)$

and the output space is one dimensional space ,$Y$ $y_i \in Y$ is *1, 2 or 3.*

## 5.11 Preprocessing

By analyzing the structure of the dataset characteristics space, it reveals that all the documents can follow some patterns. This fact will allow some preprocessing operations that will remove outliers and determine which class they belong to. The preprocessing will benefit the clustering results drastically.

### *5.11.1 Analysing the statistical thresholds to remove outliers.*

By studying the two dimensional graphics of the datasets, it is outstanding that the vast majority of the data is condensed in a small area of the space. The rest of the data, the minority of the dataset, is spread for the rest of the space in very specific ranges.

Then, it can be noticed that there is almost no text (class 1 and 2, red and blue) smaller than 6 pixels, which is quite sensible. Text fonts with smaller height than 6 are practically not used. A similar reasoning can be applied for the bigger instances of text. The use of fonts with a height larger that 55 is infrequent.

| | Range | | doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| height | [0,5] | text intances | 7,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 2,0 | 3,0 | 5,0 | 0,0 |
| | | all intances | 7,0 | 0,0 | 2,0 | 2,0 | 0,0 | 0,0 | 1,0 | 8,0 | 6,0 | 6,0 | 3,0 |
| | | % text instances inside the range | 100 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 25,0 | 50,0 | 83,3 | 0,0 |
| | [55,inf) | text intances | 11,0 | 0,0 | 1,0 | 1,0 | 1,0 | 1,0 | 0,0 | 4,0 | 0,0 | 0,0 | 0,0 |
| | | all intances | 27,0 | 18,0 | 8,0 | 8,0 | 17,0 | 17,0 | 3,0 | 4,0 | 5,0 | 10,0 | 18,0 |
| | | % text instances inside the range | 40,7 | 0,0 | 12,5 | 12,5 | 5,9 | 5,9 | 0,0 | 100 | 0,0 | 0,0 | 0,0 |
| Total number of instances of the datasets | | | 1153 | 1054 | 327 | 726 | 865 | 649 | 593 | 311 | 396 | 643 | 923 |

*Figure 41 – Number of text instances inside of the indicated ranges of height.*

Obviously, some classification errors can be done, overall because there are cases where the classification is ambiguous. A typical example would be where text is merged with graphics or images.

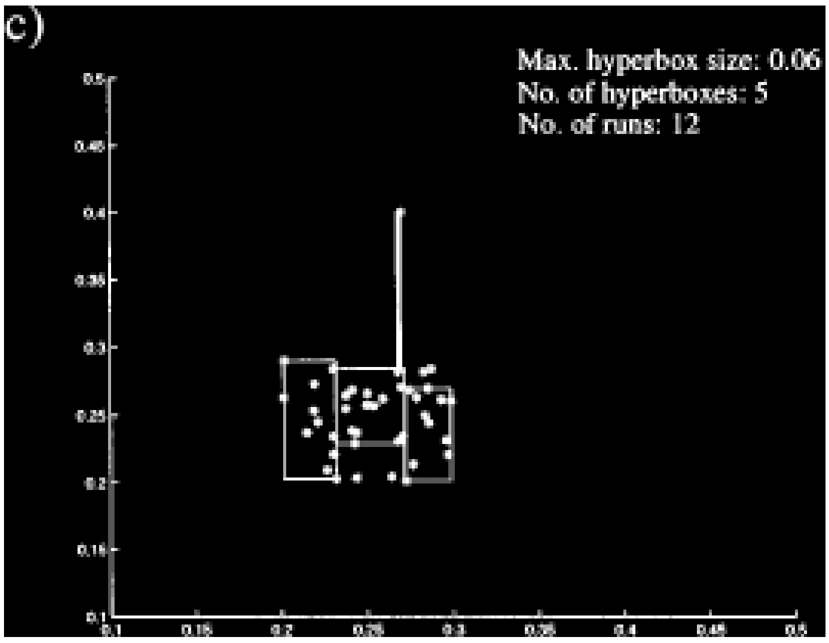


*Figure 42 – Example of an ambiguous case to classify.*

Also, it can be noticed that there is almost no text with high densities (higher than 120). It would mean that one in every two pixels of the area of the element is ‘inked’, a fact which has no sense in the habitual fonts used in text documents. At the same time there is almost no text with smaller densities than 15 because it will mean that the font is extremely thin.

| | Range | | doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| density | [120,inf) | text intances | 0,0 | 0,0 | 0,0 | 0,0 | 1,0 | 1,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| | | all intances | 1,0 | 7,0 | 0,0 | 0,0 | 10,0 | 10,0 | 6,0 | 2,0 | 7,0 | 4,0 | 19,0 |
| | | % text instances inside the range | 0,0 | 0,0 | 0,0 | 0,0 | 10,0 | 10,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| | [0,15] | text intances | 0,0 | 0,0 | 0,0 | 0,0 | 16,0 | 16,0 | 1,0 | 0,0 | 19,0 | 0,0 | 2,0 |
| | | all intances | 5,0 | 1,0 | 0,0 | 0,0 | 16,0 | 16,0 | 7,0 | 0,0 | 19,0 | 2,0 | 2,0 |
| | | % text instances inside the range | 0,0 | 0,0 | 0,0 | 0,0 | 100 | 100 | 14,3 | 0,0 | 100 | 0,0 | 100 |
| Total number of instances of the datasets | | | 1153 | 1054 | 327 | 726 | 865 | 649 | 593 | 311 | 396 | 643 | 923 |

*Figure 43 – Number of text instances inside of the indicated ranges of density.*

The elements that could get near that density values are tiny fonts or very small fonts with bold style.

GABRYS AND BARGIELA: GENERAL FUZZY MIN-MAX NEURAL NETWORK FOR CLUSTERING AND CLASSIFICATION
Wavelength λ (nm)

*Figure 44 – Examples of text instances with high density levels*

Again, it is possible to notice that there are still outliers, so a second preprocessing to remove the remaining outliers would leave a better dataset to practice clustering.

In the second preprocessing the instances in the height range of [20,55] are removed.

| | Range | | doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| height | [120,inf) | text intances | 16,0 | 2,0 | 6,0 | 6,0 | 10,0 | 10,0 | 0,0 | 50,0 | 0,0 | 0,0 | 12,0 |
| | | all intances | 19,0 | 2,0 | 7,0 | 7,0 | 10,0 | 10,0 | 0,0 | 64,0 | 0,0 | 1,0 | 13,0 |
| | | % text instances inside the range | 84,2 | 100 | 85,7 | 85,7 | 100 | 100 | | 78,1 | | 0,0 | 92,3 |
| Total number of instances of the datasets | | | 1153 | 1054 | 327 | 726 | 865 | 649 | 593 | 311 | 396 | 643 | 923 |

*Figure 45 – Number of text instances inside of the indicated ranges of height*

If the space of characteristics is reduced to instances with height in the range of [6,55] and densities in the range of [15,120], the dataset would be almost exclusively instances of the class 1 and 2.

### *5.11.2 Processing the outliers.*

We can take advantage of the previous analysis and do the first clustering to the outliers. This clustering is not based on soft computing techniques, but just on personal observations to preprocess the data.

The instances in the ranges of density [0,15] and [120, 255] and the instances in the ranges of height [0,6] and [55, ∞) will be considered as instances of the class 3 (MaDZ). The instances separated in the second preprocess, instances in the range of height [20,55] will be considered as instances of the class 2 (SZ).

## 5.12 Clustering

After preprocessing the datasets are in optimal conditions for  clustering. Different clustering techniques will be applied (and compared in the results section). The algorithms are four and belong to three families of clustering algorithms:

a) Hierarchical agglomerative clustering
b) K-means clustering
c) Prototype-based clustering. Multivariate Gaussian mixture clustering
d) Fuzzy Clustering

In the following pages, it is going to be described the fundaments of the algorithms and their variants used.

### *5.12.1 Hierarchical agglomerative clustering*

The agglomerative hierarchical cluster tree seeks to build a hierarchy of clusters. Its strategy is a "bottom up" approach, which means that each observation starts in its own cluster, and pairs of clusters are merged as one move up the hierarchy.

In order to decide which clusters should be combined, a measure of dissimilarity between sets of observations is required. In most methods of hierarchical clustering, this is achieved using appropriate metrics (a measure of distance between pairs of observations), and linkage criteria which specifies the dissimilarity of sets as a function of the pairwise distances of observations in the sets.

Given a set A of $n$ items to be clustered in $k$ clusters, the basic process of hierarchical clustering (defined by S.C. Johnson in [35]) is this:

1. It starts assigning each item to a cluster, so that if you have n items, you now have n clusters, each containing just one item. Let the distances between the clusters be the same as the distances between the items they contain.
2. Find the closest (most similar) pair of clusters and merge them into a single cluster (according to the selected linkage criterion), so that now you have one cluster less.
3. Compute distances (according to the selected metric) between the new cluster and each of the old clusters.
4. Repeat steps 2 and 3 until all items are clustered into the number of clusters *k*.

Depending on the metric and linkage criteria the result can be quite different so five metrics and seven linkage criteria will be used.

#### *5.12.1.1 Metrics*

The choice of an appropriate metric will influence the shape of the clusters, as some elements may be close to one another according to one distance and farther away according to another. For example, in a 2-dimensional space, the distance between the point (1,0) and the origin (0,0) is always 1 according to the usual norms, but the distance between the point (1,1) and the origin (0,0) can be 2, $\sqrt{2}$ or 1 under Manhattan distance, Euclidean distance or maximum distance respectively. Let *a* and *b* two instances of the dataset. The metrics used are in this project:

| ***Names*** | ***Formula*** |
|---|---|
| Euclidean distance | $\sqrt{\sum_i (a_i - b_i)^2}$ |
| Squared Euclidean distance | $\sum_i (a_i - b_i)^2$ |
| Manhattan distance [36] | $\sum_i \lvert a_i - b_i \rvert$ |
| Mahalanobis distance [36] | $\sqrt{(a-b)^T S^{-1} (a-b)}$ where S is the covariance matrix |
| Cosine distance | $\frac{a \, . \, b}{\lVert a \rVert \lVert b \rVert}$ |
| Chebyshev distance [36] | $\max_i (\lvert a_i - b_i \rvert)$ |

*Figure 46 – Table with the distance metrics used*

#### 5.12.1.2 Linkage criteria

The linkage criterion determines the distance between sets of observations as a function of the pairwise distances between observations.

Some commonly used linkage criteria between two sets of observations A and B are:

| ***Names*** | ***Formula*** |
|---|---|
| Complete linkage [38] | $\max\{d(a,b): a \in A, b \in B\}$ |
| Single linkage | $\min\{d(a,b): a \in A, b \in B\}$ |
| Average linkage [39] | $\frac{1}{\lvert A\rvert \lvert B\rvert}\sum_{a\in A}\sum_{b\in B} d(a,b)$ |
| Centroid linkage | $\lVert \overline{x_A} - \overline{x_B} \rVert_2$ where $\overline{x_K} = \frac{1}{n_k}\sum_{i=0}^{n_k} x_{ki}$ |
| Ward linkage [37] | $n_A\, n_B \frac{\lVert \overline{x_A} - \overline{x_B} \rVert_2^2}{n_A + n_B}$ where $\overline{x_K} = \frac{1}{n_k}\sum_{i=0}^{n_k} x_{ki}$ |
| weighted linkage | $d\big((a,b),c\big) = \frac{d(a,c)+d(b,c)}{2}$ where C is a third observed set |
| median linkage | Similar to average linkage but using the statistical median |

*Figure 47 – Table with the linkage criteria used.*

### 5.12.2 K-means clustering [40]

K-means (MacQueen, 1967) is one of the simplest unsupervised learning algorithms to solve clustering problems. The procedure classifies a given dataset through a certain number of clusters (assume k clusters) fixed a priori.

It defines k centroids, one for each cluster. These centroids shoud be placed in a cunning way because of different location causes different result. So, the best choice is to place them as much far away as possible from each other. In the next step it takes each instance belonging to a given dataset and it associates it to the nearest centroid. When no instance is pending, the first step is completed and an early groupage is done. At these instances the algorithm re-calculates k new centroids as barycenters of the clusters resulting from the previous step. After calculating these k new centroids, a new binding has to be done between the data and the nearest new centroid. Thus, a loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words, centroids do not move any more.

Finally, this algorithm aims at minimizing an objective function, in this case, a squared error function. The objective function

$$,J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$

where $\left\| x_i^{(j)} - c_j \right\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre $c_j$ , is an indicator of the distance of the n data points from their respective cluster centres.

The algorithm is composed of the following steps:

1. Place k points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the k centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

Although it can be proved that the procedure will always terminate, the k-means algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. The algorithm is also significantly sensitive to the initial randomly selected cluster centres. The k-means algorithm can be run multiple times to reduce this effect.

### 5.12.3 Multivariate Gaussian mixture clustering and Expectation Maximization [41]

In practice, each cluster can be mathematically represented by a parametric distribution, like a Gaussian or Normal distribution. The entire data set is therefore modelled by a mixture of these distributions. An individual distribution used to model a specific cluster is often referred to as a component distribution.

- A mixture model with high likelihood tends to have the following features:
- component distributions have high "peaks" (data in one cluster are tight);
- the mixture model "covers" the data well (dominant patterns in the data are captured by component distributions).

Main advantages of model-based clustering:

- well-studied statistical inference techniques available;
- flexibility in choosing the component distribution;
- obtaining of a density estimation for each cluster;
- a "soft" classification is available.

The multivariate normal distribution of a n-dimensional random vector x

$$X = [x_1, x_2, \ldots, x_n]$$

can be written in the following notation

$$X \sim N_n(\mu, \Sigma)$$

with *n*-dimensional mean vector

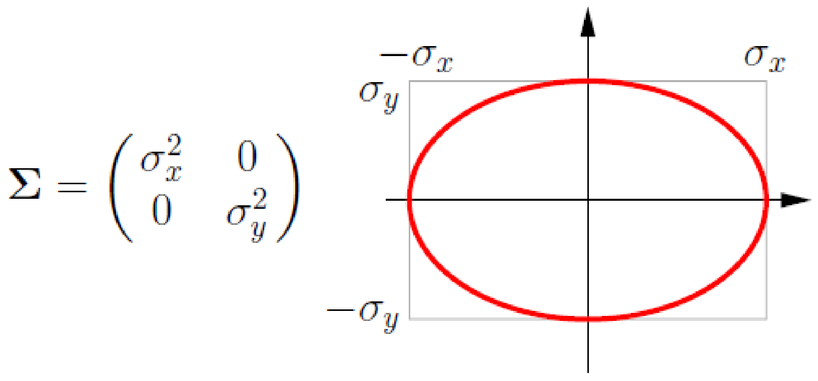
$$\mu = [E[x_1], \ldots, E[x_n]]$$

and $n_x n$ covariance matrix

$$\Sigma = [Cov[x_i, x_j]], \quad with \;\; i,j = 1,2,\ldots,n$$

Regarding the covariance matrix it can be considered two cases:

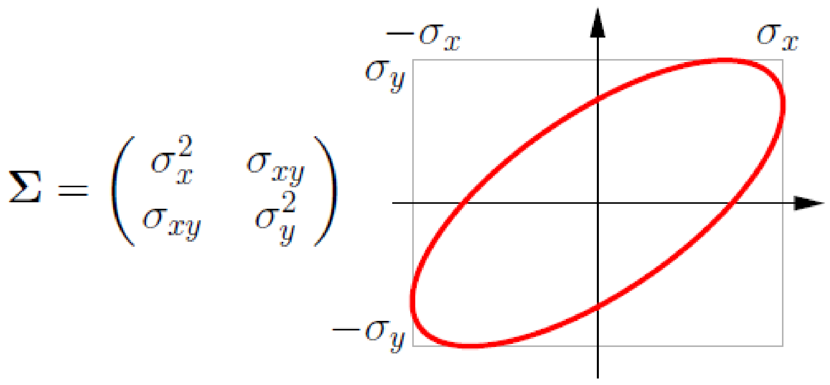
the covariance matrix is diagonal. The contour lines are axes-parallel ellipses.



$\Sigma = \begin{pmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{pmatrix}$
$-\sigma_x$
$\sigma_x$
$\sigma_y$
$-\sigma_y$

Figure 48 – Graphical example of the case of a diagonal covariance matrix

the covariance matrix is full. The contour lines are rotated ellipses.



$\Sigma = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix}$
$-\sigma_x$
$\sigma_x$
$\sigma_y$
$-\sigma_y$

Figure 49 – Graphical example of the case of a full covariance matrix

Consequence: A full covariance matrix describes a scaling and a rotation.

#### 5.12.3.1 Gaussian Mixture Models

Like K-Means, Gaussian Mixture Models (GMM) can be regarded as a type of unsupervised learning or clustering methods. They are among the most statistically mature methods for clustering. But unlike K-Means, GMMs (visit [42] for more details) are able to build soft clustering boundaries, i.e., points in space can belong to any class with a given probability. Actually, clusters can be considered as Gaussian distributions centred on their barycentres, as we can see in this picture, where the grey circle represents the first variance of the distribution.

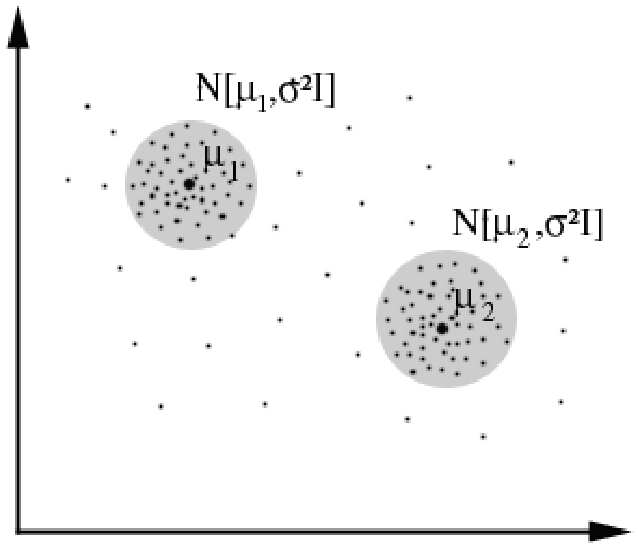


Figure 50 – Graphical example of the first variance of the two Gaussian distributions

In statistics, a mixture model is a probabilistic model which assumes the underlying data to belong to a mixture distribution. In a mixture distribution, its density function is just a convex combination (a linear combination in which all coefficients or weights sum to one) of other probability density functions:

$$p(x) = w_1 p_1(x) + \cdots + w_n p_n(x)$$

The individual $p_i(x)$ density functions that are combined to make the mixture density $p(x)$ are called the mixture components, and the weights $w_1, w_2, \ldots, w_n$ associated with each component are called the mixture weights or mixture coefficients.

The most common mixture distribution is the Gaussian (Normal) density function, in which each of the mixture components are Gaussian distributions, each of them with their own mean and variance parameters.

$$p(x) = w_1 N(x|\mu_1 \Sigma_1) + \cdots + w_n N(x|\mu_n \Sigma_n)$$

We can obtain the likelihood of the sample:

$$P(x|w_i, \mu_1, \ldots, \mu_n)$$

What we really want to maximize is

$$P(x|\mu_1, \dots, \mu_n)$$

the probability of a datum, given the centers of the Gaussians

$$P(x|\mu_i) = \sum_i P(w_i)P(x|w_i, \mu_1, \dots, \mu_n)$$

It is the base to write the likelihood function:

$$P(data|\mu_i) = \prod_{i=i}^{k} \sum_i P(w_i)P(x|w_i, \mu_1, \dots, \mu_n)$$

Now we should maximize the likelihood function by calculating $\frac{\partial L}{\partial \mu_i} = 0$, but it would be too difficult. That's why we use a simplified algorithm called EM (Expectation-Maximization).

#### *5.12.3.2 The EM Algorithm*

The algorithm which is used in practice to find the mixture of Gaussians that can model the data set is called EM (Expectation-Maximization) (Dempster, Laird and Rubin, 1977).

In statistics, an expectation–maximization (EM) algorithm is an iterative method for finding maximum likelihood of parameters in statistical models, where the model depends on unobserved latent variables. The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters; and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step.

Given a statistical model consisting of a set *X* of observed data, a set of unobserved latent data or missing values *Z,* and a vector of unknown parameters $\theta$, along with a likelihood function $L(\theta; X, Z) = p(X, Z|\theta)$ , the maximum likelihood estimate (MLE) of the unknown parameters is determined by the marginal likelihood of the observed data

$$L(\theta; X) = p(X|\theta) = \sum_Z p(X, Z|\theta)$$

However, this quantity is often intractable.

The EM algorithm seeks to find the MLE of the marginal likelihood by iteratively applying the following two steps:

Expectation step (E-step): Calculate the expected value of the log likelihood function, with respect to the conditional distribution of $Z$ given $X$ under the current estimate of the parameters $\theta^{(t)}$:

$$Q(\theta|\theta^{(t)}) = E_{Z|X,\theta^{(t)}}[log\, L(\theta; X, Z)]$$

Maximization step (M-step): Find the parameter that maximizes this quantity:

$$\theta^{(t+1)} = argmax_{\theta}\, Q(\theta|\theta^{(t)})$$

### *5.12.3.3 The EM Algorithm applied to the Gaussian mixture models*

Let $X = [x_1, \ldots, x_n]$ be a sample of n independent observations from a mixture of two multivariate normal distributions of dimension d, and let $z=(z_1,\ldots,z_n)$ be the latent variables that determine the component from which the observation originates.

For more detailed information the reader is addressed to [44].

$$X_i|(Z_i = 1) \sim N_d(\mu_1, \Sigma_1) \;\; and \; X_i|(Z_i = 2) \sim N_d(\mu_2, \Sigma_2)$$

Where

$$P|(Z_i = 1) = \tau_1 \;\; and \; P|(Z_i = 2) = \tau_2 = 1 - \tau_1$$

The aim is to estimate the unknown parameters representing the "mixing" value between the Gaussians and the means and covariances of each:

$$\theta = (\tau, \mu_1, \mu_2 \Sigma_{1,} \Sigma_2)$$

where the likelihood function is:

$$L(\theta; x, z) = \; p(x, z|\theta) = \prod_{i=1}^{n} \sum\nolimits_{j=1}^{2} \mathbb{I}(z_i = j)\, \tau_j f(x_i; \mu_{j,} \Sigma_j)$$

where $\mathbb{I}$ is an indicator function and f is the probability density function of a multivariate normal. This may be rewritten in exponential family form:

$$L(\theta; x, z) = \left\{ \sum_{i=1}^{n} \sum\nolimits_{j=1}^{2} \mathbb{I}(z_i = j)\, [log\tau_j - \frac{1}{2} log|\Sigma_j| - \frac{1}{2}(x_i - \mu_j)^T \Sigma_j^{-1}(x_i - \mu_j) - \frac{d}{2} log\,(2\pi)] \right\}$$

To see the last equality, note that for each i all indicators $\mathbb{I}(z_i = j)$ are equal to zero, except for one which is equal to one. The inner sum thus is reduced to a single term.

***E-step***

Given our current estimate of the parameters *θ(t),* the conditional distribution of the $Z_i$ is determined by Bayes theorem to be the proportional height of the normal density weighted by *τ:*

$$T_{j,i}^{(t)}: P\left(Z_i = j \middle| X_i = x_i; \theta^{(t)}\right) = \frac{\tau_j^{(t)} f(x_i; \mu_j^{(t)}, \Sigma_j^{(t)})}{\tau_1^{(t)} f\left(x_i; \mu_1^{(t)}, \Sigma_1^{(t)}\right) + \tau_2^{(t)} f(x_i; \mu_2^{(t)}, \Sigma_2^{(t)})}$$

Thus, the E step results in the function:

$$Q\left(\theta \middle| \theta^{(t)}\right) = E_{Z|X,\theta^{(t)}}[log\, L(\theta; X, Z)] =$$

$$= \sum_{i=1}^{n} \sum\nolimits_{j=1}^{2} T_{j,i}^{(t)} \left[log\tau_j - \frac{1}{2} log|\Sigma_j| - \frac{1}{2}(x_i - \mu_j)^T \Sigma_j^{-1}(x_i - \mu_j) - \frac{d}{2} log\,(2\pi)\right]$$

***M-step***

The quadratic form of $Q\left(\theta \middle| \theta^{(t)}\right)$ means that determining the maximising values of θ is relatively straightforward. Firstly, note that *τ, (μ₁,Σ₁)* and *(μ₂,Σ₂)* may be all maximised independently of each other since they all appear in separate linear terms.

Firstly, consider τ, which has the constraint *$\tau_1 + \tau_2 = 1$:*

$$\tau^{(t+1)} = argmax_\tau\, Q\left(\theta \middle| \theta^{(t)}\right) = argmax_\tau \left\{\left[\sum_{i=1}^{n} T_{1,i}^{(t)}\right] log\tau_1 + \left[\sum_{i=1}^{n} T_{2,i}^{(t)}\right] log\tau_2\right\}$$

This has the same form as the MLE for the binomial distribution, so:

$$\tau_j^{(t+1)} = \frac{\sum_{i=1}^{n} T_{j,i}^{(t)}}{\sum_{i=1}^{n}(T_{1,i}^{(t)} + T_{2,i}^{(t)})} = \frac{1}{n} \sum\nolimits_{i=1}^{n} T_{j,i}^{(t)}$$

For the next estimates of $(\mu_1, \Sigma_1)$:

$$(\mu_1^{(t+1)}, \Sigma_1^{(t+1)}) = argmax_{\mu_1 \Sigma_1}\, Q\left(\theta \middle| \theta^{(t)}\right) =$$

$$= argmax_{\mu_1 \Sigma_1} \sum\nolimits_{i=1}^{n} T_{1,i}^{(t)} \left\{-\frac{1}{2} log\tau_1 |\Sigma_1| - -\frac{1}{2}(x_i - \mu_1)^T \Sigma_1^{-1}(x_i - \mu_1)\right\}$$

This has the same form as a weighted MLE for a normal distribution, so

$$\mu_1^{(t+1)} = \frac{\sum_{i=1}^{n} T_{1,i}^{(t)} x_i}{\sum_{i=1}^{n} T_{1,i}^{(t)}}$$

and

$$\Sigma_1^{(t+1)} = \frac{\sum_{i=1}^{n} T_{1,i}^{(t)} \left(x_i - \mu_1^{(t+1)}\right)\left(x_i - \mu_1^{(t+1)}\right)^T}{\sum_{i=1}^{n} T_{1,i}^{(t)}}$$

and, by symmetry:

$$\mu_2^{(t+1)} = \frac{\sum_{i=1}^{n} T_{2,i}^{(t)} x_i}{\sum_{i=1}^{n} T_{2,i}^{(t)}}$$

and

$$\Sigma_2^{(t+1)} = \frac{\sum_{i=1}^{n} T_{2,i}^{(t)} \left(x_i - \mu_1^{(t+1)}\right)\left(x_i - \mu_1^{(t+1)}\right)^T}{\sum_{i=1}^{n} T_{2,i}^{(t)}}$$

### *5.12.4 Fuzzy k-means clustering*

Fuzzy c-means (FCM) is a method of clustering which allows the instances of the dataset to belong to two or more clusters. This method (developed by Dunn in [45] and improved by Bezdek in [46]) is based on minimization of the following objective function:

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^m \left\| x_i - c_j \right\|^2 , \qquad 1 \le m < \infty$$

where $m$ is any real number greater than 1, $u_{ij}$ is the degree of membership of $x_i$ in the cluster $j$, $x_i$ is the $i^{th}$ of $d$-dimensional measured data, $c_j$ is the $d$-dimension center of the cluster, and ||*|| is any norm expressing the similarity between any measured data and the center.

The parameter $m$ is called the fuzziness coefficient; the higher $m$ is, the softer the cluster boundaries are.

Fuzzy partitioning is carried out through an iterative optimization of the objective function shown above, with the update of membership $u_{ij}$ and the cluster centers $c_j$ by:

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \frac{\left\| x_i - c_j \right\|}{\left\| x_i - c_j \right\|}^{\frac{2}{m-1}}}$$

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^m x_i}{\sum_{i=1}^{N} u_{ij}^m}$$

This iteration will stop when $max_{ij} \left\{ \left| u_{ij}^{(k+1)} - u_{ij}^{(k)} \right| \right\} < \varepsilon$, where $\varepsilon$ is a termination criterion between 0 and 1, whereas $k$ are the iteration steps.

This procedure converges to a local minimum or a saddle point of $J_m$. The algorithm is composed of the following steps:

1. Initialize $U$=[$u_{ij}$] matrix, $U^{(0)}$
2. At $k$-step: calculate the centers vectors $C^{(k)}$=[$c_j$] with $U^{(k)}$

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^m x_i}{\sum_{i=1}^{N} u_{ij}^m}$$

3. Update $U^{(k)}$ , $U^{(k+1)}$

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \frac{\|x_i - c_j\|}{\|x_i - c_j\|}^{\frac{2}{m-1}}}$$

4. If $\| U^{(k+1)} - U^{(k)} \| < \varepsilon$ then STOP; otherwise return to step 2.

As already told, data are bound to each cluster by means of a Membership Function, which represents the fuzzy behaviour of this algorithm. To do that, we simply have to build an appropriate matrix named U whose factors are numbers between 0 and 1, and represent the degree of membership between data and centers of clusters.

Finally, to classify every instance of the data as MiDZ or SZ it is necessary to discretize memberships values to 1 or 0.

# 6 Experiments

In order to clarify the experimentation process the experiments carried out will try to find out the best clustering technique of the previously studied.

## 6.1 Agglomerative hierarchical clustering

There are different metric distances and linkages involved, but not all the combinations between them are possible or have sense. The cases studied are:

| | LINKAGE CRITERIA | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| DISTANCE METRICS | | AVERAGE | COMPLETE | SINGLE | WEIGHTED | WARD | MEDIAN | CENTROID |
| | EUCLIDEAN | X | X | X | X | X | X | X |
| | SQ. EUCLIDEAN | X | X | X | X | X | X | X |
| | MANHATTAN | X | X | X | X | | | |
| | MINKOSKI e. 0.5 | X | X | X | X | | | |
| | MINKOSKI e. 3 | X | X | X | X | | | |
| | CHEVYCHEV | X | X | X | X | | | |
| | MAHALANOBIS | X | X | X | X | | | |
| | COSINE | X | X | X | X | | | |

*Figure 51 – Table of the considered combinations between the different distance metrics and linkage criteria*

## 6.2 K-means clustering

For the K-means clustering can be considered different distance metrics. The cases studied are three: squared Euclidean distance, Manhattan distance and cosine distance

## 6.3 Multivariate Gaussian mixture models clustering

Regarding multivariate Gaussian mixture models clustering the options compared are two. The case where the covariance matrix is diagonal and the case where is full.

## 6.4 Fuzzy k means

Fuzzy K-means clustering allow some parametric control with the fuzziness coefficient, so different values will be experimented: 2, 3, 5, 10, 100 and 250.

# 7 Results

The results of every experiment done with the eleven gathered datasets are graphically and parametrically described in the following points.

All the following graphical representations of the results section are referred to the dataset created from document 1. Remember that $x$ axis is the height of a recognized element and $y$ represent its density. Red dots are text MaDZ, blue dots are SZ and green dots are MiDZ.

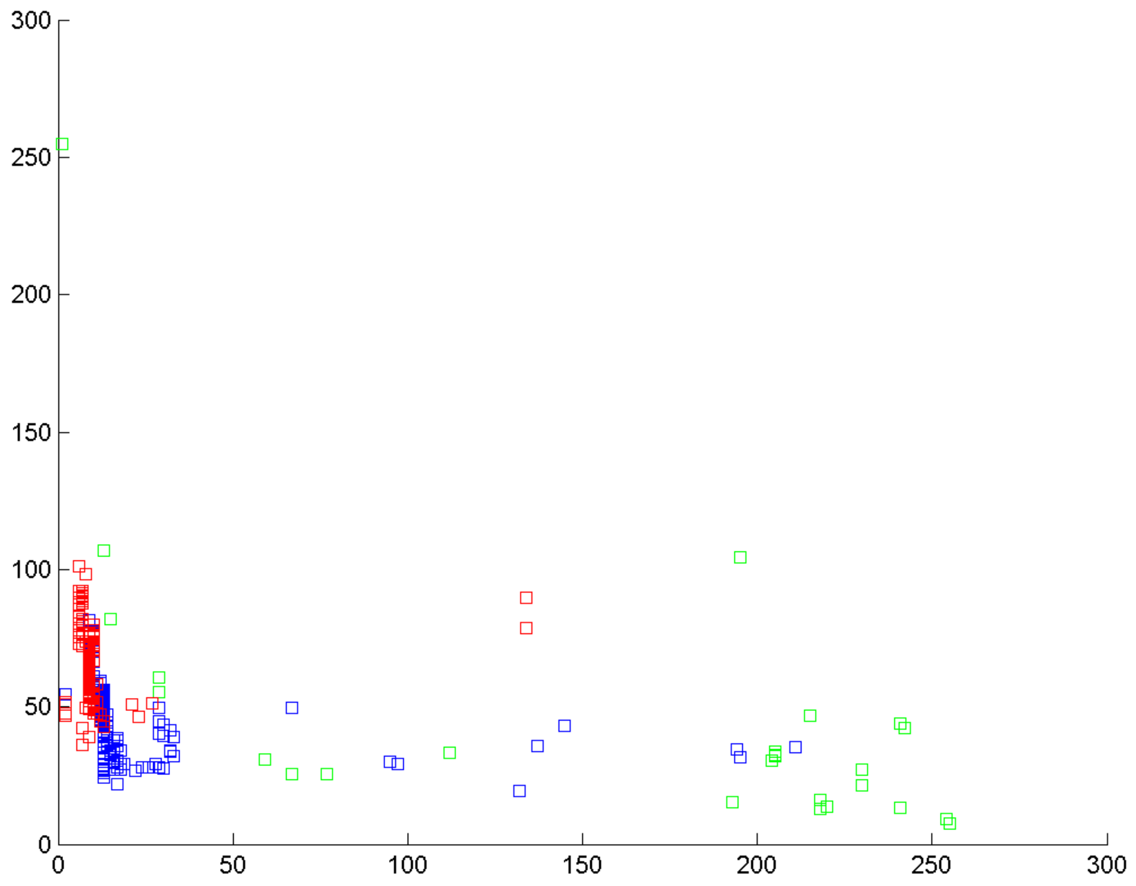
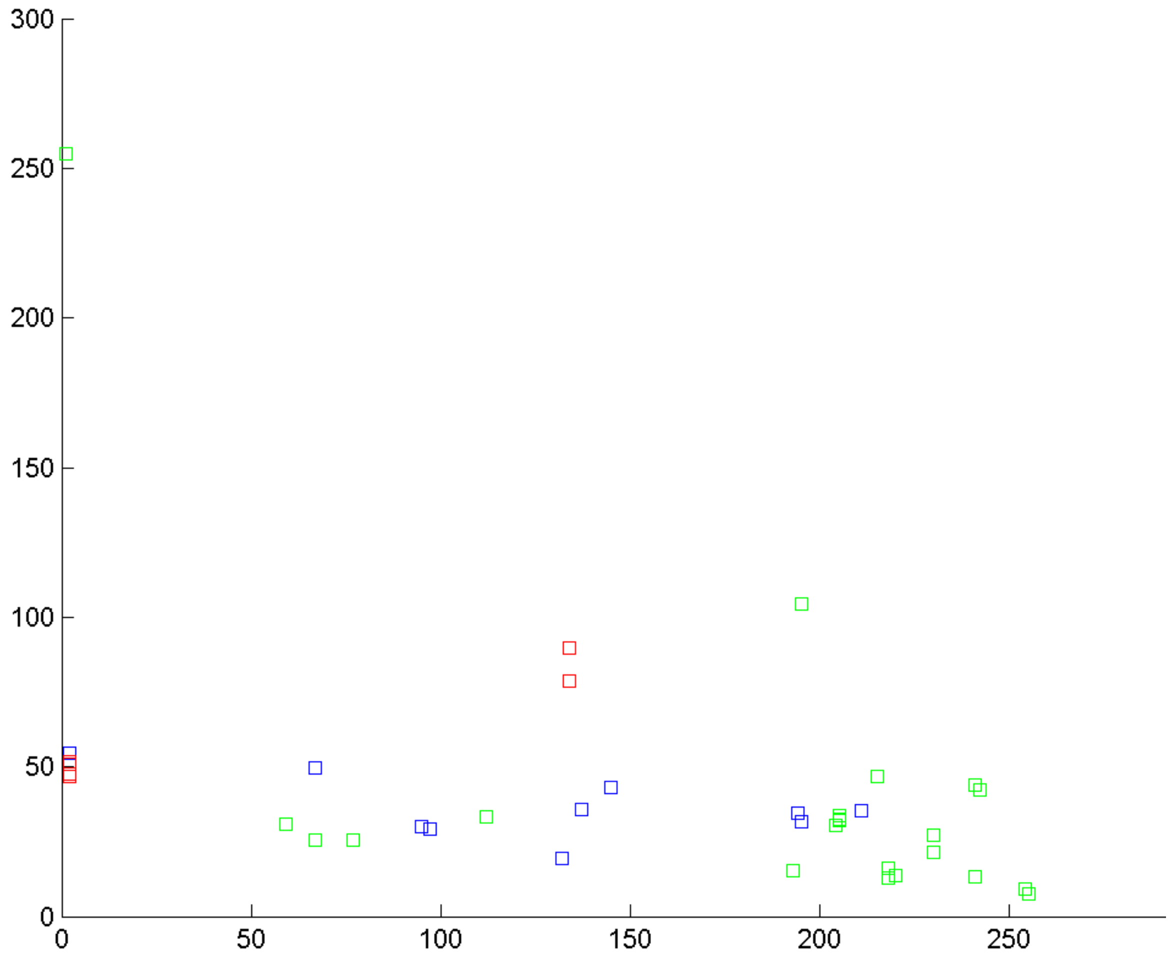


*Figure 52 – Space of characteristics of the original preprocess*

## 7.1 Preprocessing 1

As it was explained before, the first preprocessing deals with some outliers of the datasets and classifies them by following some previously described criteria. The table shows the error rate of the preprocessing 1.

$$error\ rate\ (\%) = \frac{number\ of\ instances\ classified\ wrongly}{total\ number\ of\ instances}\ x\ 100$$



| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 45 | 0 | 10 | 10 | 41,86 | 41,86 | 11,76 | 42,86 | 59,46 | 22,73 | 4,76 |

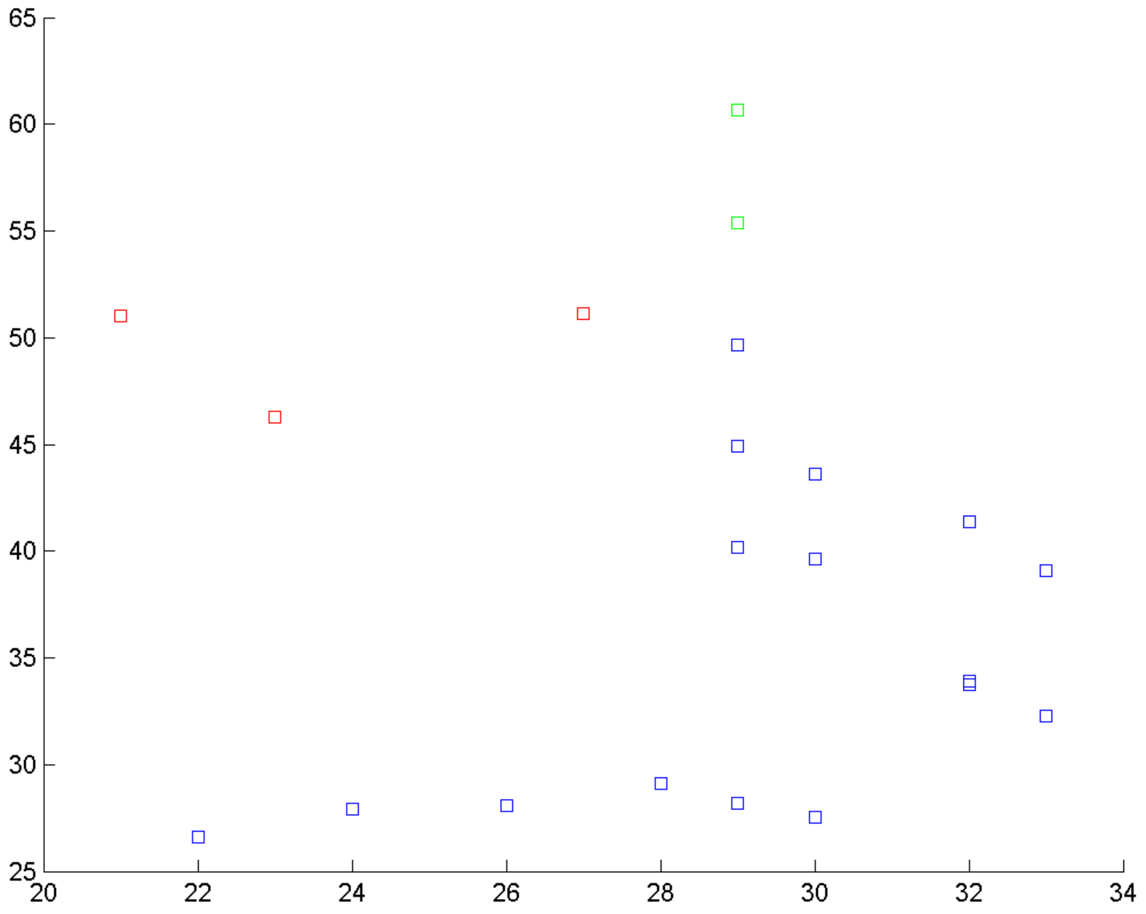
*Figure 53 –Characteristics space and error rates % associated with prepro. 1.*

The error rate can be high in some datasets but, as we will see later, it doesn't have an appreciable influence in the final error rate of the whole dataset due to the small amount of instances involved in this preprocessing.

In this methodology the error rates higher than 50% have sense because, in the preprocessing processes, the classification is done by a criterion previously establish. It is not the result of applying a soft computing technique.

## 7.2 Preprocessing 2

The second preprocessing deals with the rest of outliers of the datasets and classifies them by following additional criteria also explained in previous sections. The table shows the error rate of the preprocessing 2.



| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 23,81 | 71,43 | 14,29 | 14,29 | 28,57 | 28,57 | 0 | 30,56 | 0 | 0 | 42,86 |

*Figure 54 –Characteristics space and error rates % associated with prepro. 2.*

Again, the error rate can be high and, in some cases, unsatisfactory but, as the first preprocessing, it doesn’t have an appreciable influence in the final error rate.

## 7.3 Preprocessing 1 and 2 results. Summary table

The following table shows the minimal influence of the mistakes done in both preprocessings in the final error rate of the whole dataset.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Error rate (%) from preprocessing 1 | 45,0 | 0,0 | 10,0 | 10,0 | 41,9 | 41,9 | 11,8 | 42,9 | 59,5 | 22,7 | 4,8 |
| Error rate (%) from preprocessing 2 | 23,8 | 71,4 | 14,3 | 14,3 | 28,6 | 28,6 | 0 | 30,6 | 0 | 0 | 42,9 |
| Error rate (%) from both preprocessings together | 37,7 | 15,2 | 11,8 | 11,8 | 38,6 | 38,6 | 5,3 | 32,6 | 31,5 | 12,0 | 17,5 |
| Number of instances for both preprocessings | 61 | 33 | 17 | 17 | 57 | 57 | 32 | 86 | 39 | 25 | 63 |
| Total number of instances of the datasets | 1153 | 1054 | 327 | 726 | 865 | 649 | 593 | 311 | 396 | 643 | 923 |
| Percentage of preprocessed instances | 5,3 | 3,1 | 5,2 | 2,3 | 6,6 | 8,8 | 5,4 | 27,7 | 9,8 | 3,9 | 6,8 |
| Error rate (%) from both preprocessings together | 0,02 | 0,00 | 0,01 | 0,00 | 0,03 | 0,03 | 0,03 | 0,09 | 0,06 | 0,01 | 0,01 |

*Figure 55 – Table with the percentage of errors by preprocessing and their influence in the total percentage of errors.*

## 7.4 Agglomerative hierarchical clustering

Using agglomerative hierarchical clustering the comparison would be from different distance metrics and linkage criteria. For the following points, the tables will show the ratio of the mistakes done by the algorithms with a different formula. They will be calculated in the positive way, it means, in terms of success rate.

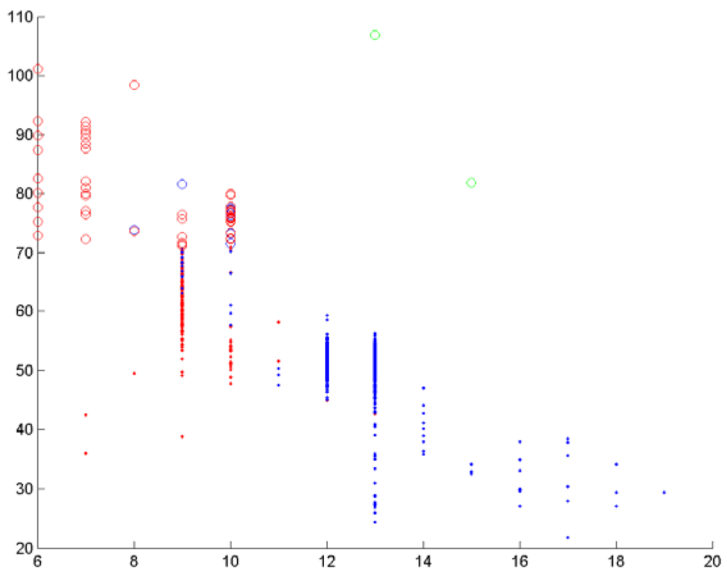
$$success\ rate\ (\%) = \frac{number\ of\ instances\ classified\ correctly}{total\ number\ of\ instances}\ x\ 100$$

$$success\ rate = 100 - \ error\ rate$$

All the following graphical representations of the results section are referred to the dataset created from document 1.

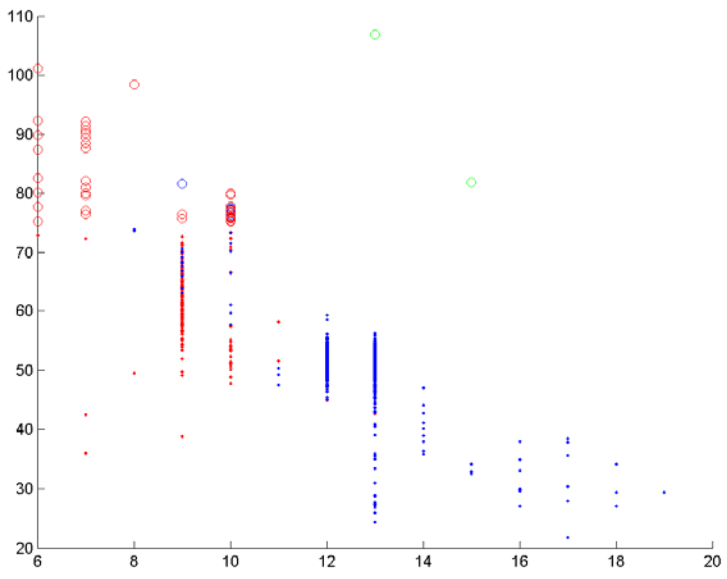
The graphical representations and the success rates tables are:

### 7.4.1 Euclidean distance and average linkage



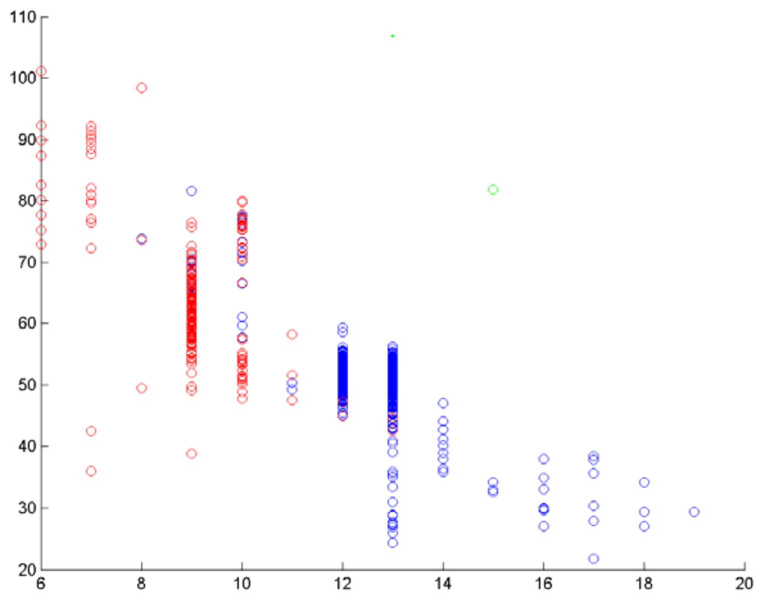
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 82,97 | 81,68 | 81,29 | 70,66 | 75,25 | 65,54 | 70,77 | 77,23 | 48,46 | 33,98 | 75,23 |

### 7.4.2 Euclidean distance and complete



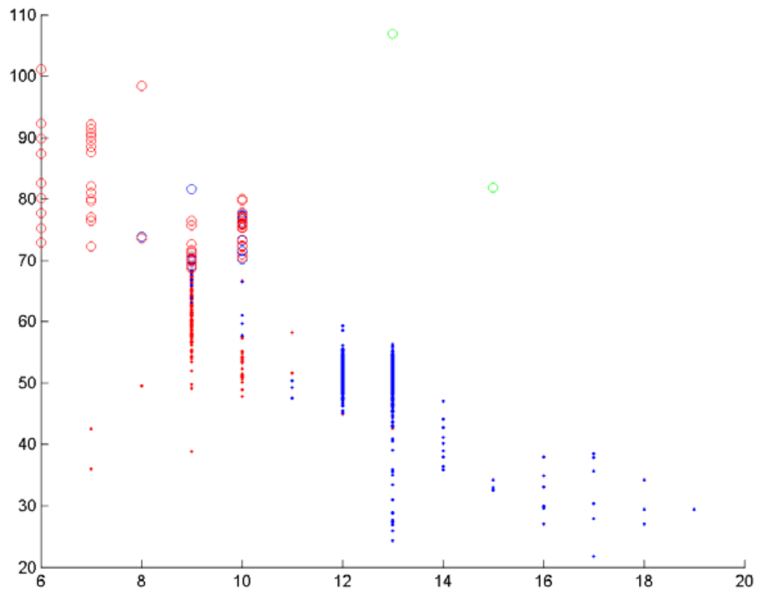
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 81,87 | 82,86 | 81,29 | 70,24 | 75,50 | 65,71 | 69,52 | 82,59 | 54,06 | 67,96 | 77,09 |

### 7.4.3 Euclidean distance and single linkage



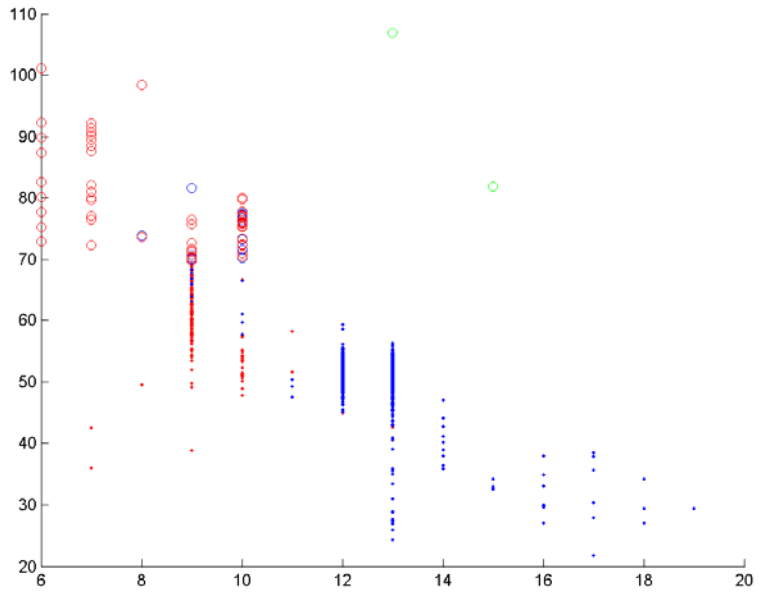
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 21,79 | 81,00 | 81,29 | 70,66 | 73,27 | 65,71 | 70,77 | 83,04 | 49,86 | 33,01 | 75,00 |

### 7.4.4 Euclidean distance and weighted linkage



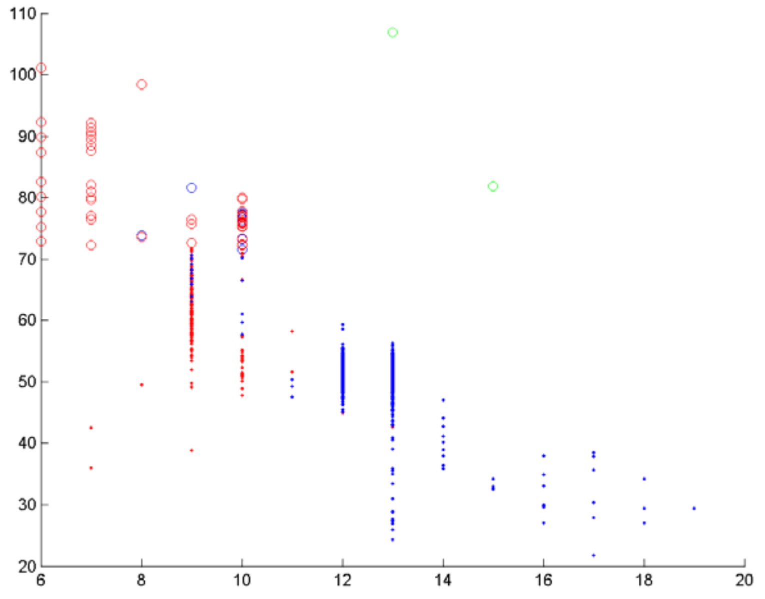
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 83,61 | 57,98 | 81,29 | 77,01 | 72,40 | 65,71 | 70,23 | 82,59 | 48,46 | 66,83 | 73,02 |

### 7.4.5 Euclidean distance and ward linkage



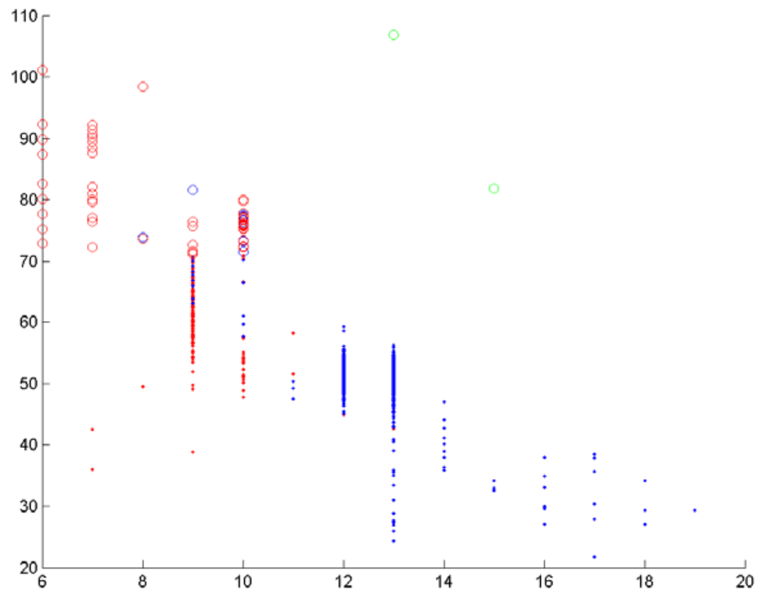
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 83,42 | 84,13 | 81,29 | 79,55 | 74,13 | 69,76 | 69,52 | 77,23 | 53,78 | 33,98 | 77,09 |

### 7.4.6 Euclidean distance and median linkage



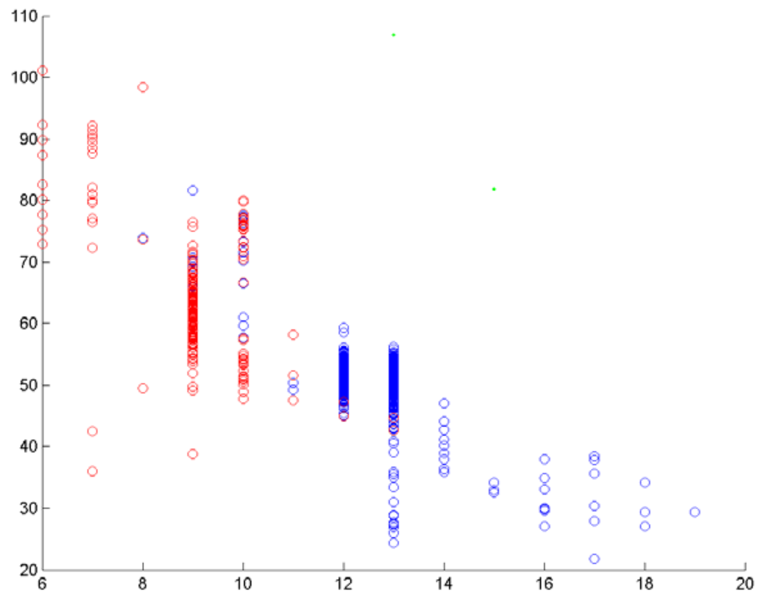
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 82,69 | 84,04 | 81,29 | 69,82 | 74,88 | 66,22 | 71,84 | 82,59 | 48,46 | 61,49 | 60,58 |

### 7.4.7 Euclidean distance and centroid linkage



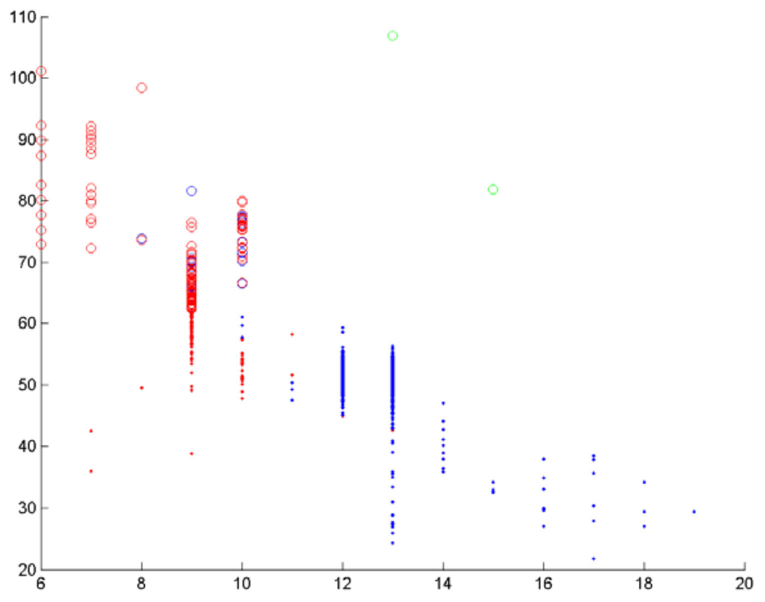
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 76,19 | 28,57 | 85,71 | 85,71 | 71,43 | 71,43 | 0,00 | 69,44 | 0,00 | 0,00 | 57,14 |

### 7.4.8 Squared Euclidean distance and average linkage



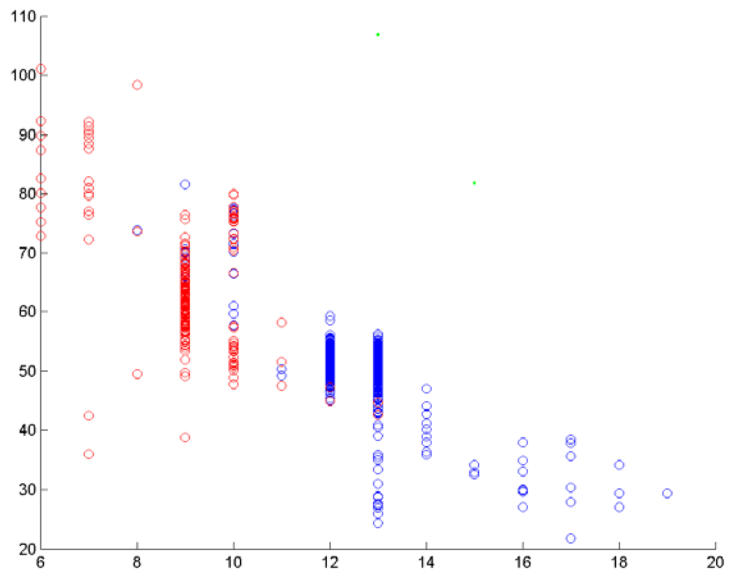
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 82,97 | 81,68 | 81,29 | 70,66 | 75,25 | 65,54 | 70,77 | 77,23 | 48,46 | 33,98 | 75,23 |

### 7.4.9 Squared Euclidean distance and complete



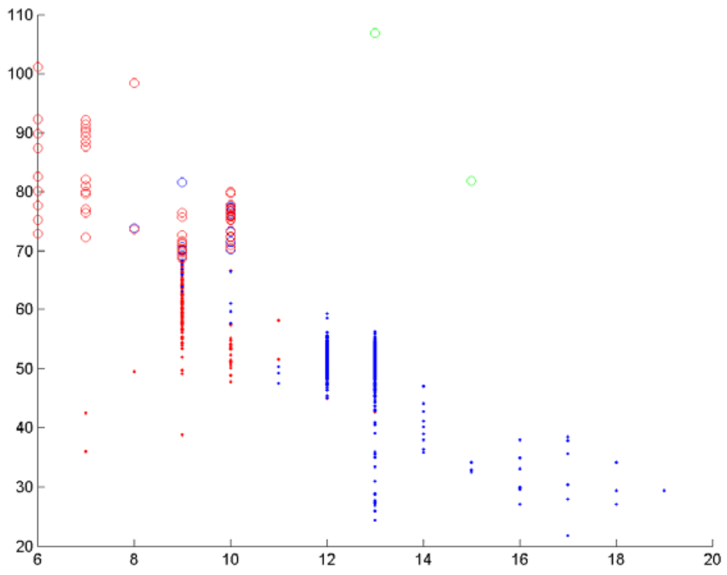
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 81,87 | 82,86 | 81,29 | 70,24 | 75,50 | 65,71 | 69,52 | 82,59 | 54,06 | 67,96 | 77,09 |

### 7.4.10 Squared Euclidean distance and single linkage



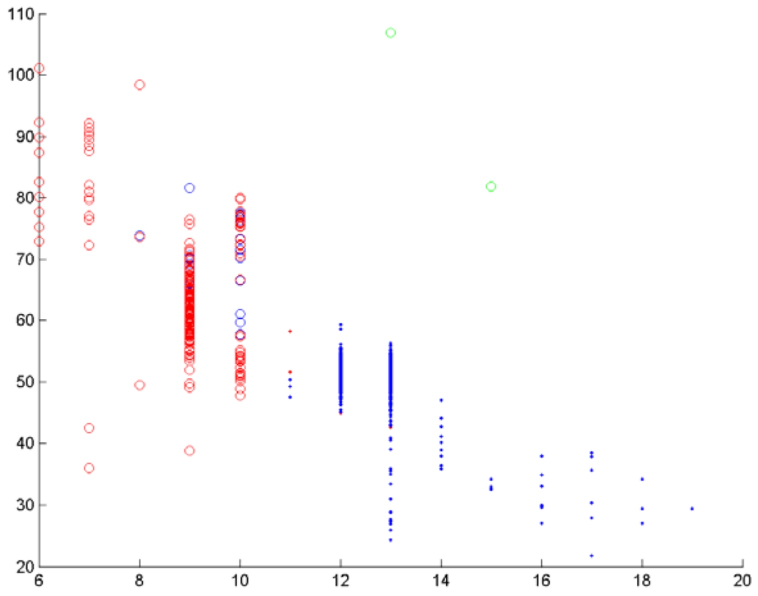
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 21,79 | 81,00 | 81,29 | 70,66 | 73,27 | 65,71 | 70,77 | 83,04 | 49,86 | 33,01 | 75,00 |

### 7.4.11 Squared Euclidean distance and weighted linkage



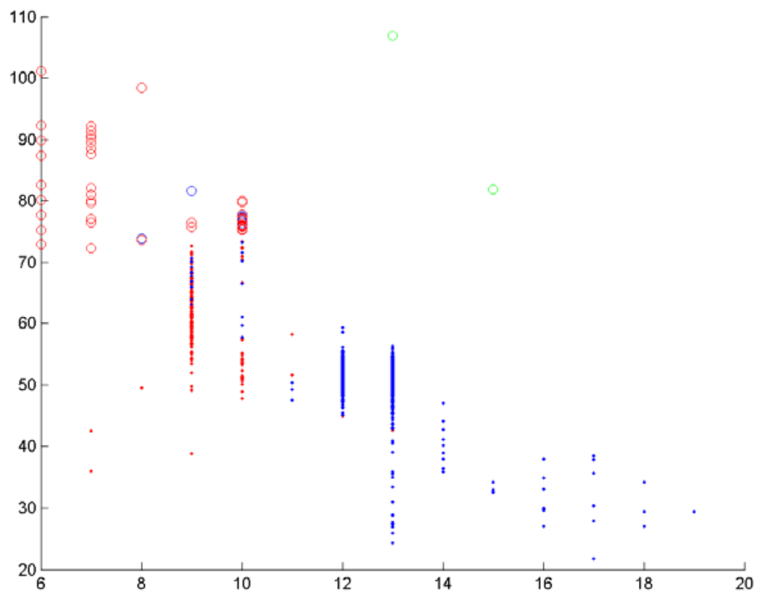
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 83,61 | 57,98 | 81,29 | 77,01 | 72,40 | 65,71 | 70,23 | 82,59 | 48,46 | 66,83 | 73,02 |

### 7.4.12 Squared Euclidean distance and ward linkage



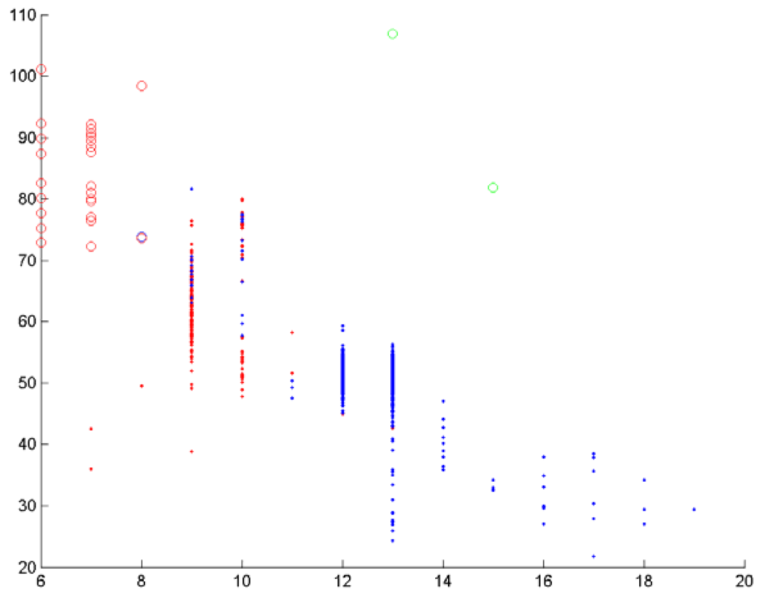
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 83,42 | 84,13 | 81,29 | 79,55 | 74,13 | 69,76 | 69,52 | 77,23 | 53,78 | 33,98 | 77,09 |

### 7.4.13 Squared Euclidean distance and median linkage



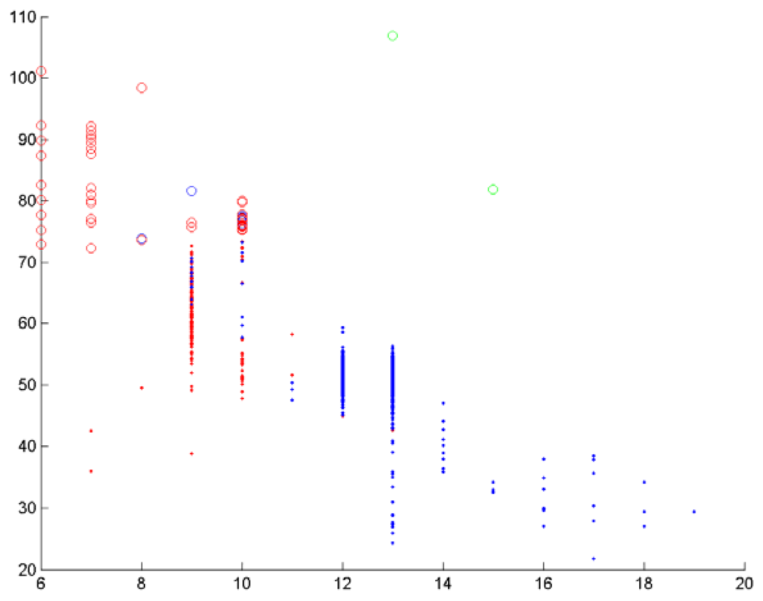
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 82,69 | 84,04 | 81,29 | 69,82 | 74,88 | 66,22 | 71,84 | 82,59 | 48,46 | 61,49 | 60,58 |

### 7.4.14 Squared Euclidean distance and centroid linkage



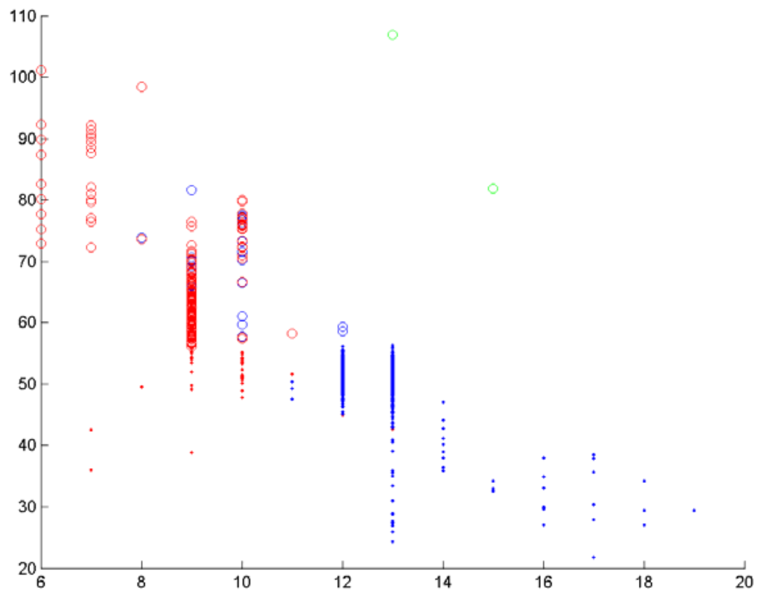
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 76,19 | 28,57 | 85,71 | 85,71 | 71,43 | 71,43 | 0,00 | 69,44 | 0,00 | 0,00 | 57,14 |

### 7.4.15 Manhattan distance and average linkage



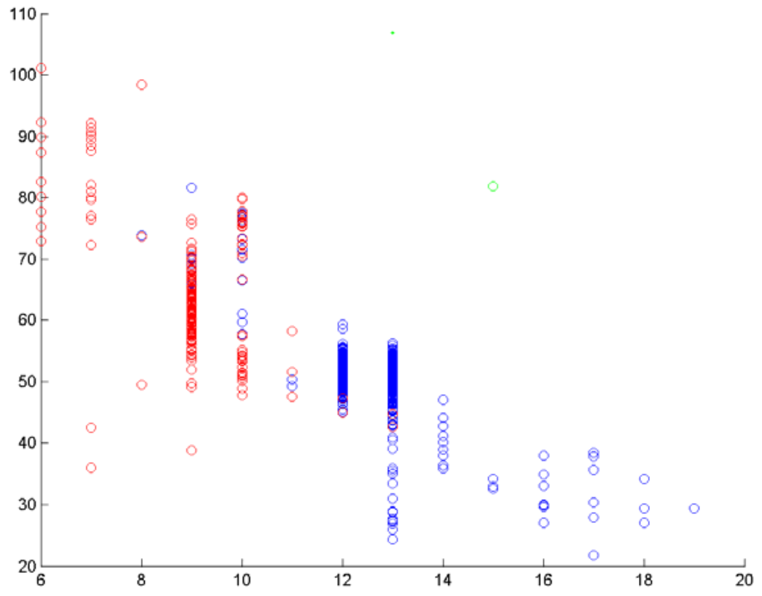
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 82,97 | 81,68 | 81,29 | 70,66 | 75,25 | 65,54 | 70,77 | 77,23 | 48,46 | 33,98 | 75,23 |

### 7.4.16 Manhattan distance and complete



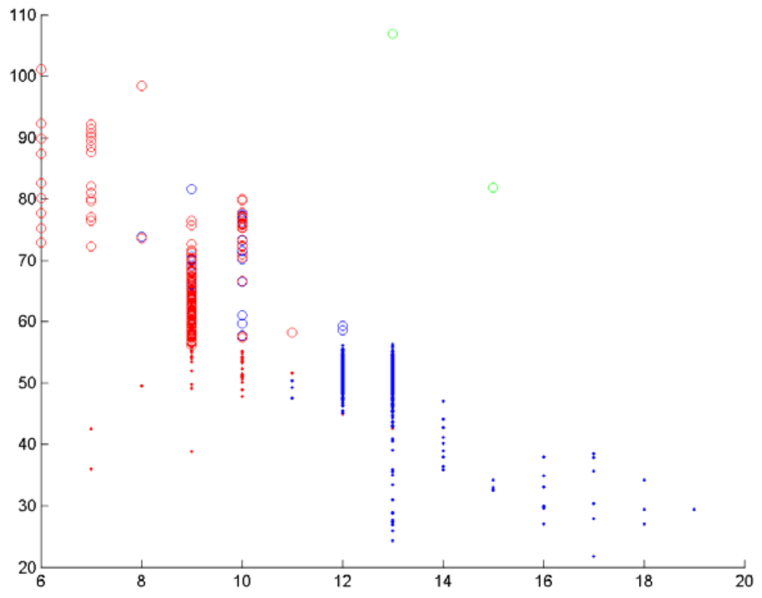
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 81,87 | 82,86 | 81,29 | 70,24 | 75,50 | 65,71 | 69,52 | 82,59 | 54,06 | 67,96 | 77,09 |

### 7.4.17 Manhattan distance and single linkage



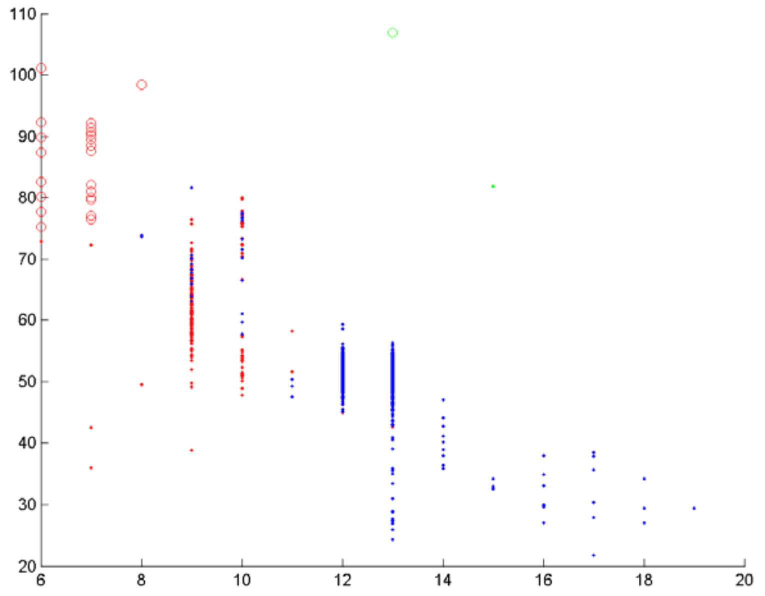
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 82,42 | 81,19 | 81,29 | 70,66 | 74,88 | 65,54 | 70,77 | 78,57 | 48,46 | 33,98 | 75,23 |

### 7.4.18 Manhattan distance and weighted linkage



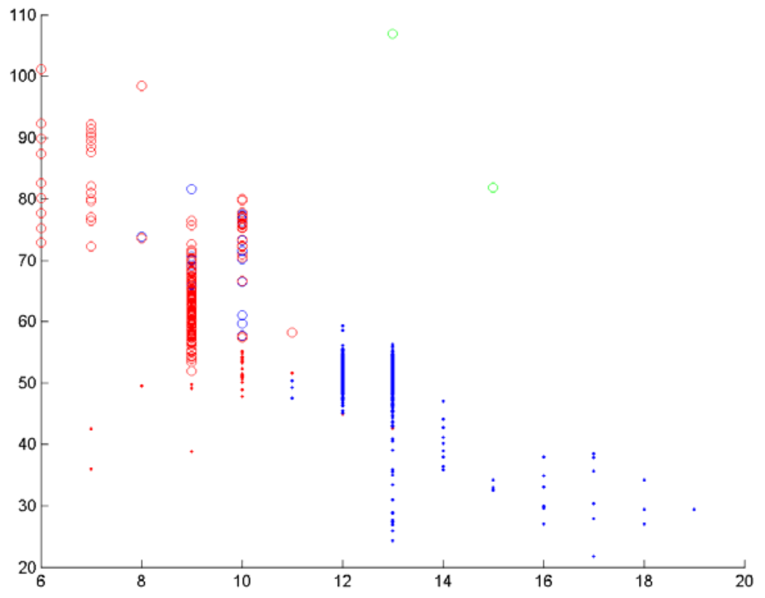
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 93,32 | 81,49 | 81,29 | 69,82 | 75,50 | 66,39 | 70,77 | 77,23 | 48,46 | 33,82 | 74,88 |

### 7.4.19 Minkowski (exp 0.5) distance and average linkage



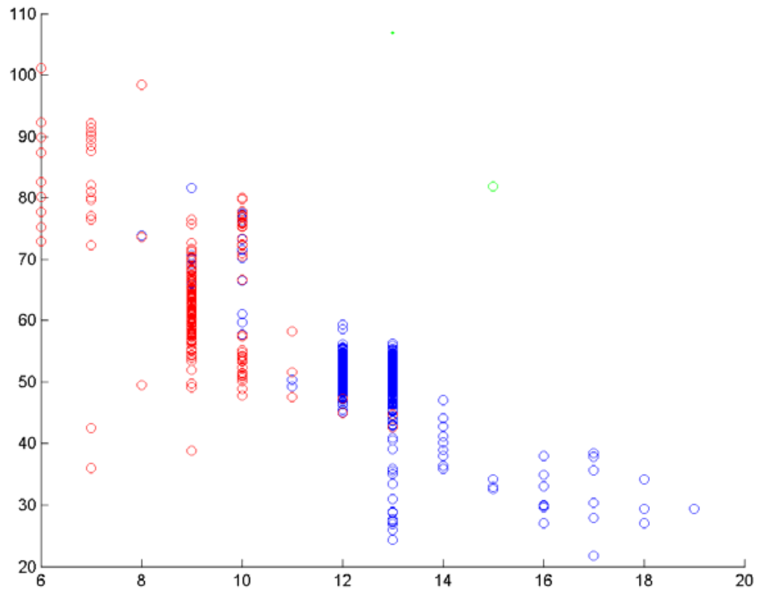
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 81,23 | 81,00 | 81,29 | 79,27 | 74,88 | 65,54 | 70,77 | 77,23 | 48,46 | 33,82 | 74,88 |

### 7.4.20 Minkowski (exp 0.5) distance and complete



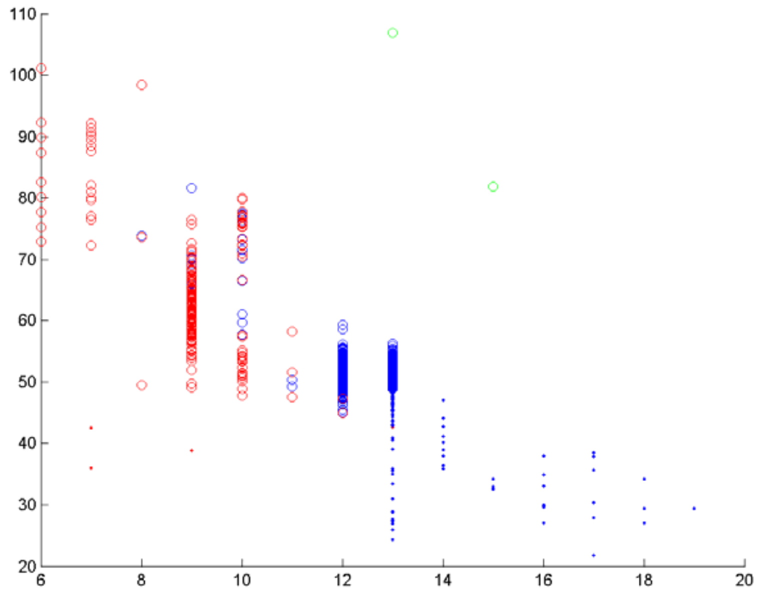
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 94,41 | 84,43 | 81,29 | 74,47 | 81,44 | 66,39 | 70,23 | 77,23 | 51,26 | 33,82 | 76,05 |

### 7.4.21 Minkowski (exp 0.5) distance and single linkage



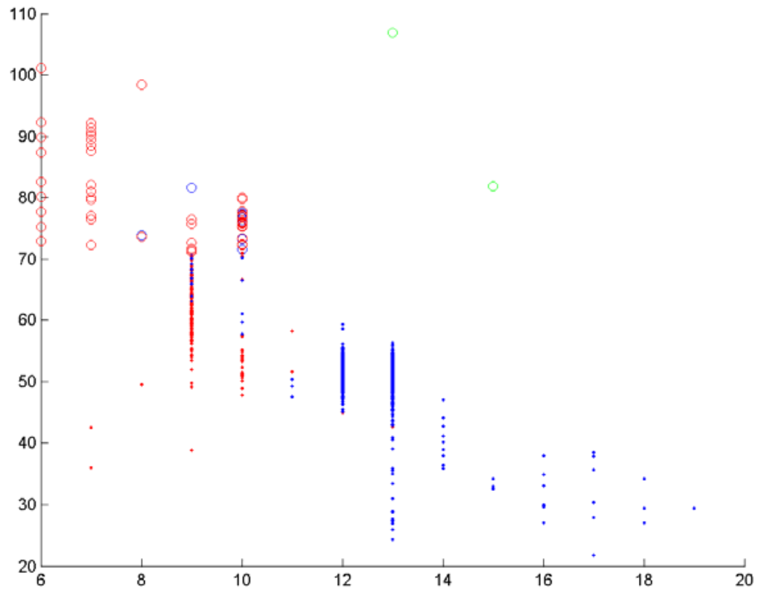
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 21,79 | 81,00 | 80,32 | 70,24 | 73,27 | 65,71 | 70,77 | 83,04 | 49,86 | 33,01 | 75,23 |

### 7.4.22 Minkowski (exp 0.5) distance and weighted linkage



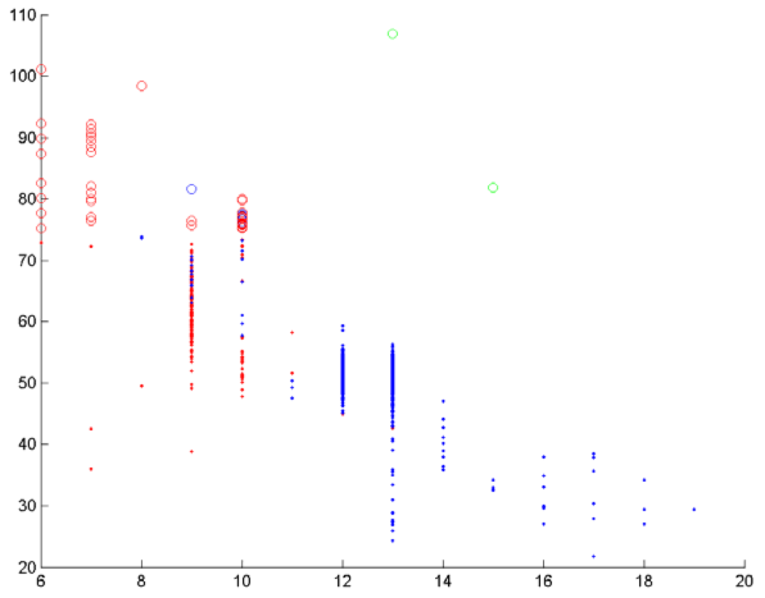
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 30,86 | 83,64 | 81,29 | 72,50 | 75,37 | 65,71 | 70,23 | 83,04 | 51,26 | 33,82 | 77,21 |

### 7.4.23 Minkowski (exp 3) distance and average linkage



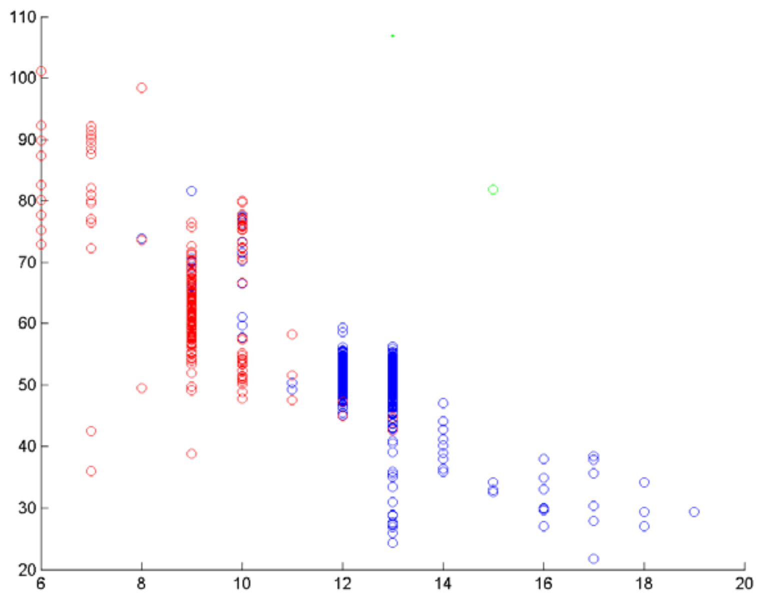
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 82,97 | 81,68 | 81,29 | 70,66 | 75,25 | 65,54 | 70,77 | 77,23 | 48,46 | 33,98 | 75,23 |

### 7.4.24 Minkowski (exp 3) distance and complete



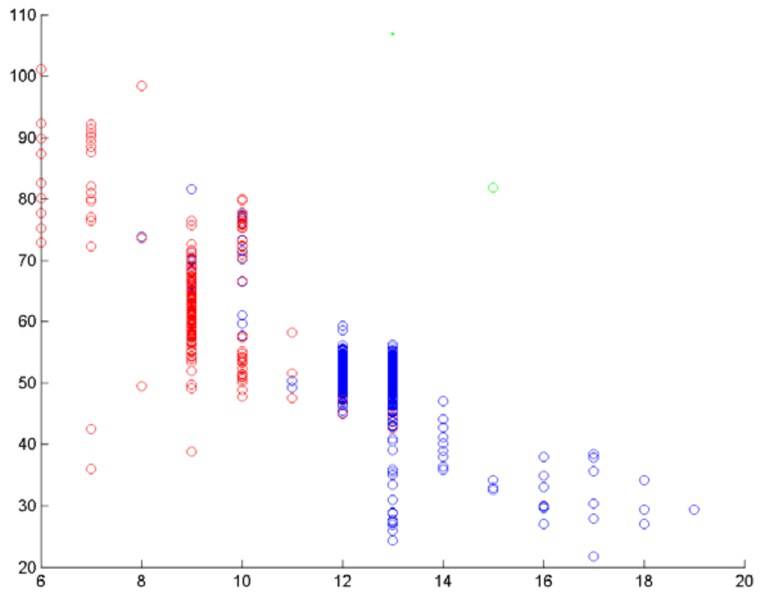
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 81,87 | 82,86 | 81,29 | 70,24 | 75,50 | 65,71 | 69,52 | 82,59 | 54,06 | 67,96 | 77,09 |

### 7.4.25 Minkowski (exp 3) distance and single linkage



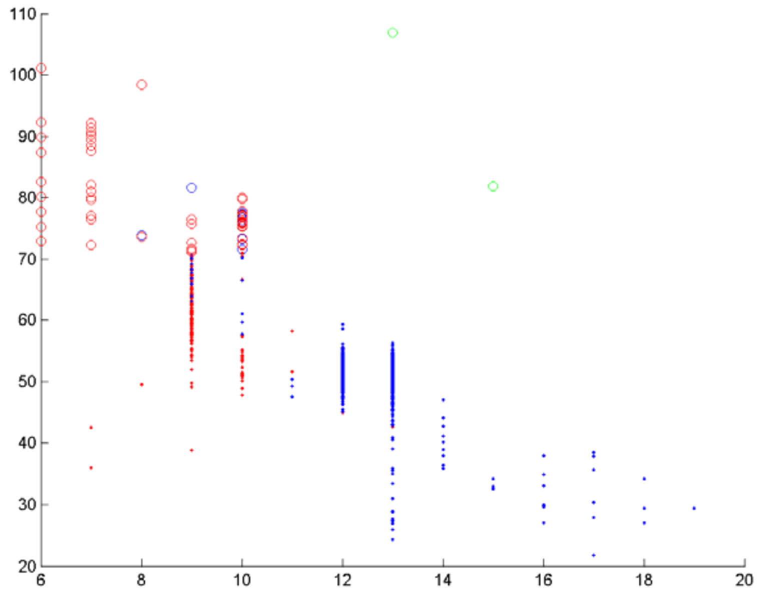
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 21,79 | 81,00 | 81,29 | 70,66 | 73,27 | 65,71 | 70,77 | 83,04 | 49,86 | 33,01 | 75,00 |

### 7.4.26 Minkowski (exp 3) distance and weighted linkage



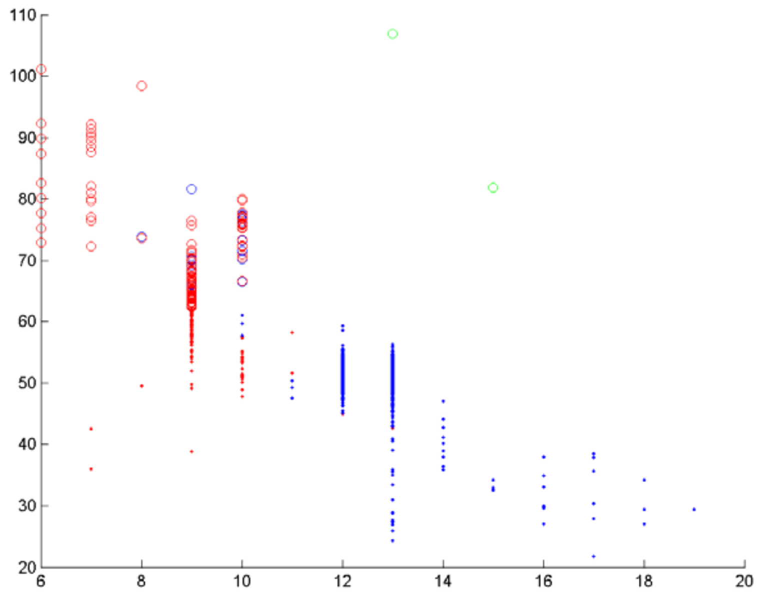
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 83,61 | 57,98 | 81,29 | 77,01 | 72,40 | 65,71 | 70,23 | 82,59 | 48,46 | 66,83 | 73,02 |

### 7.4.27 Chebychev distance and average linkage



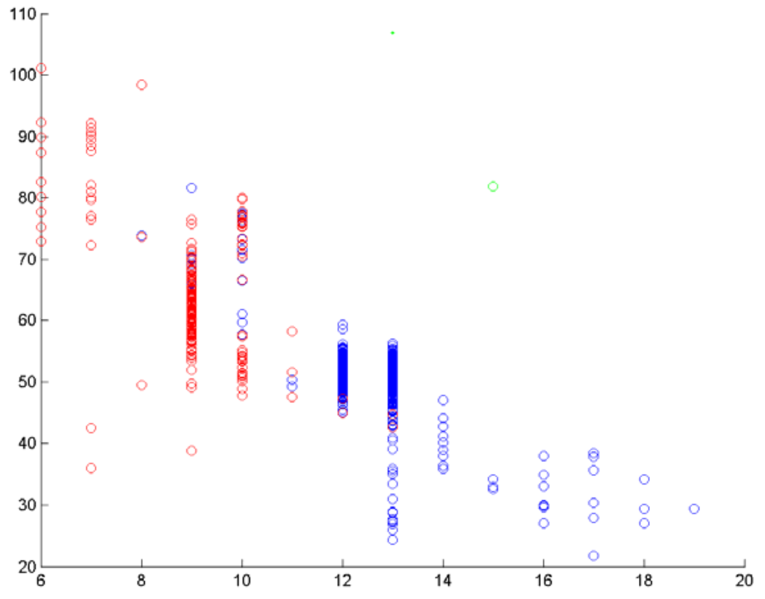
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 82,97 | 81,68 | 81,29 | 70,66 | 75,25 | 65,54 | 70,77 | 77,23 | 48,46 | 33,98 | 75,23 |

### 7.4.28 Chebychev distance and complete



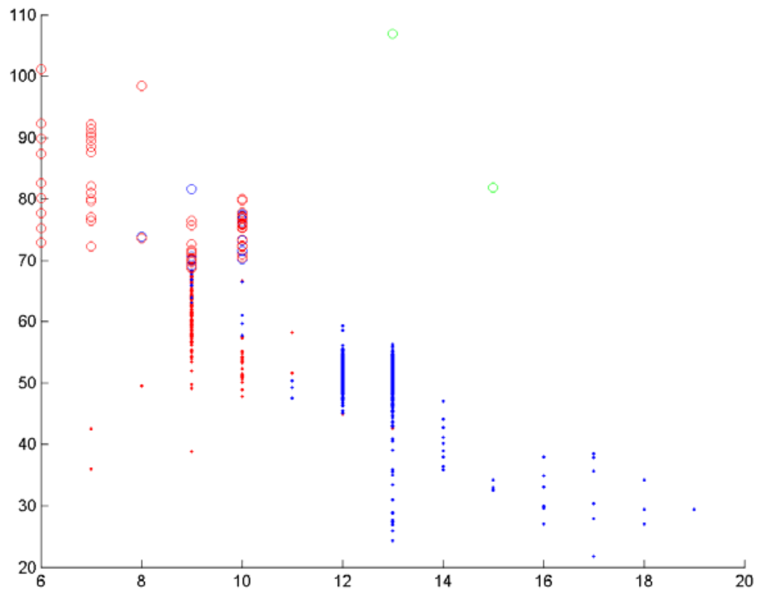
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 81,87 | 82,86 | 81,29 | 70,24 | 75,50 | 65,71 | 69,52 | 82,59 | 54,06 | 67,96 | 77,09 |

### 7.4.29 Chebychev distance and single linkage



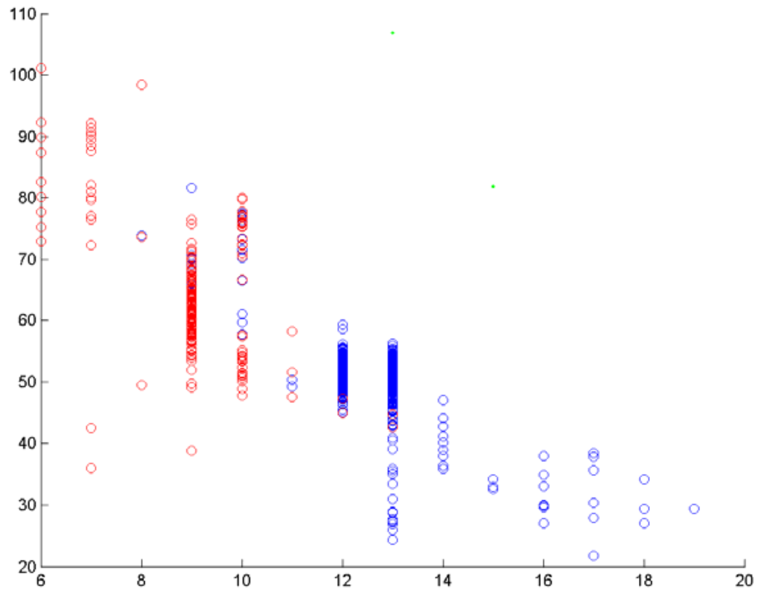
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 21,79 | 81,00 | 81,29 | 70,66 | 73,27 | 65,71 | 70,77 | 83,04 | 49,86 | 33,01 | 75,00 |

### 7.4.30 Chebychev distance and weighted linkage



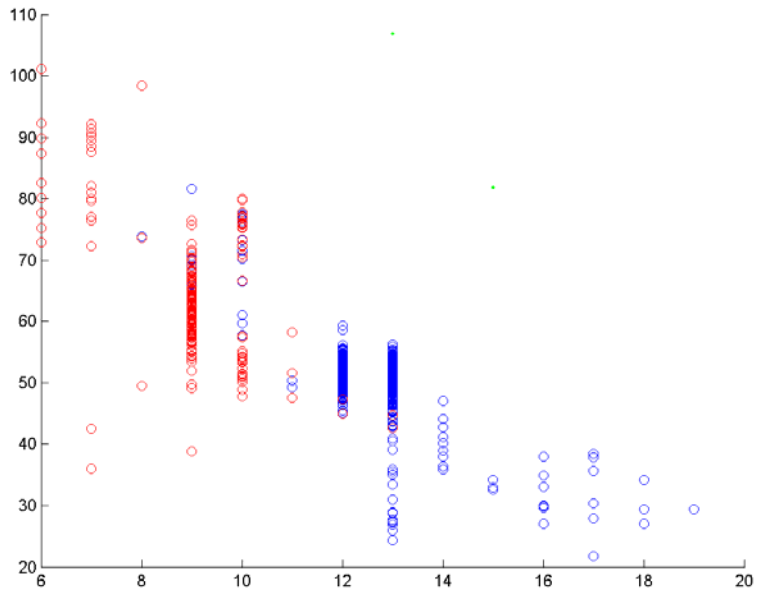
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 83,61 | 57,98 | 81,29 | 77,01 | 72,40 | 65,71 | 70,23 | 82,59 | 48,46 | 66,83 | 73,02 |

### 7.4.31 Mahalanobis distance and average linkage



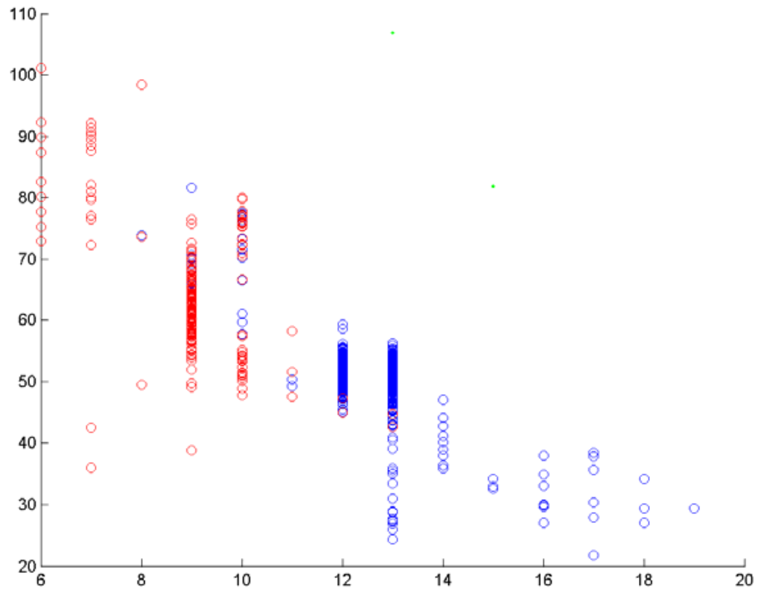
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 82,97 | 81,68 | 81,29 | 70,66 | 75,25 | 65,54 | 70,77 | 77,23 | 48,46 | 33,98 | 75,23 |

### 7.4.32 Mahalanobis distance and complete



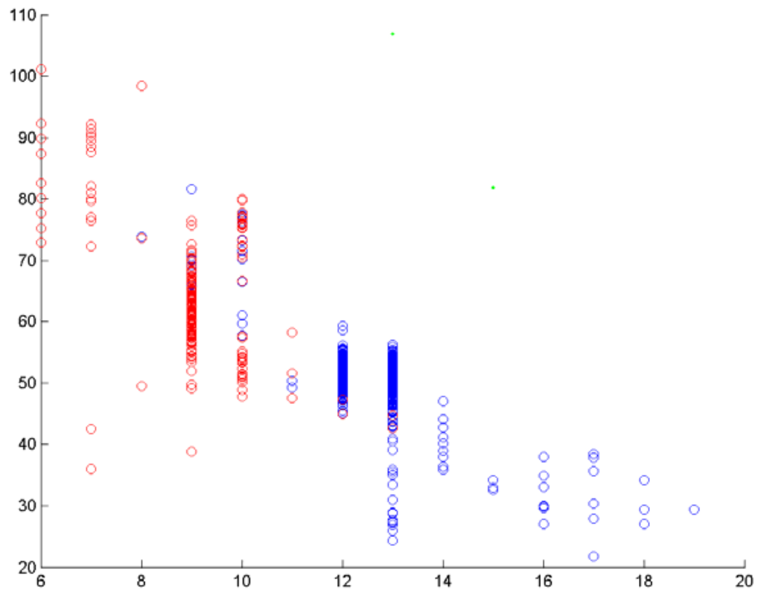
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 81,87 | 82,86 | 81,29 | 70,24 | 75,50 | 65,71 | 69,52 | 82,59 | 54,06 | 67,96 | 77,09 |

### 7.4.33 Mahalanobis distance and single linkage



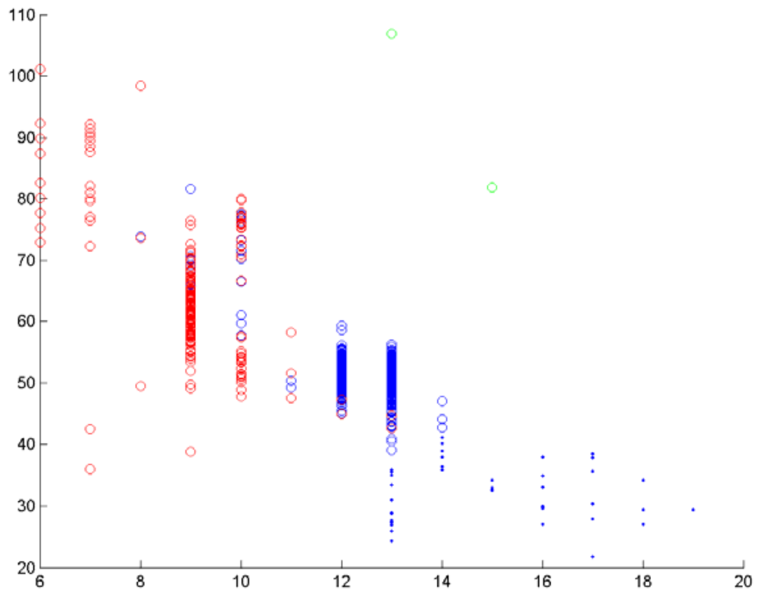
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 21,79 | 81,00 | 81,29 | 70,66 | 73,27 | 65,71 | 70,77 | 83,04 | 49,86 | 33,01 | 75,00 |

### 7.4.34 Mahalanobis distance and weighted linkage



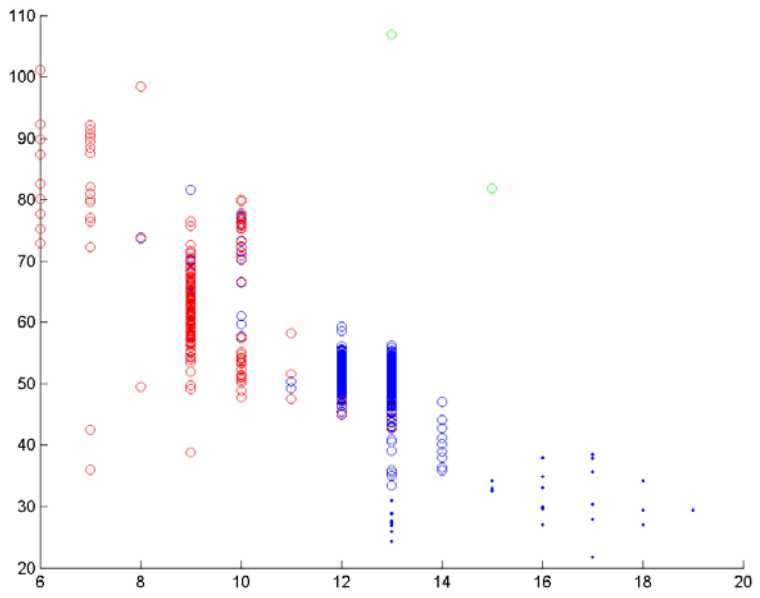
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 83,61 | 57,98 | 81,29 | 77,01 | 72,40 | 65,71 | 70,23 | 82,59 | 48,46 | 66,83 | 73,02 |

### 7.4.35 Cosine distance and average linkage



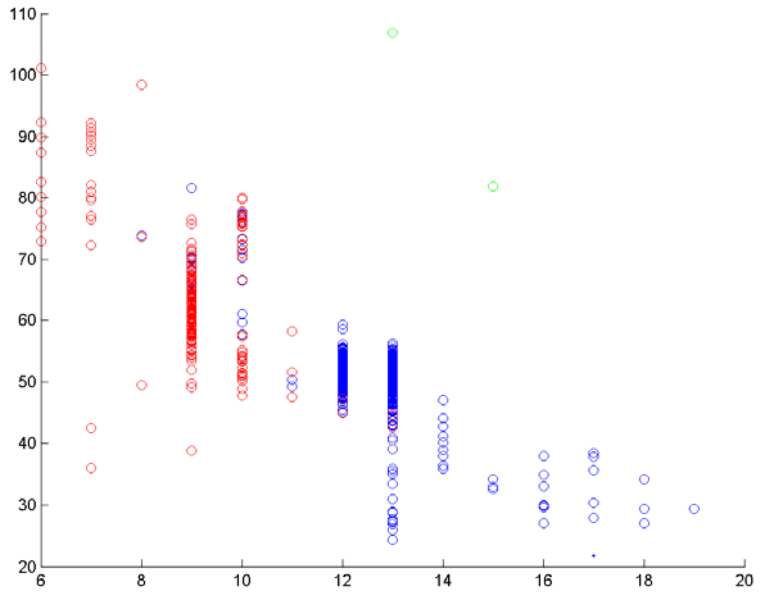
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 82,97 | 81,68 | 81,29 | 70,66 | 75,25 | 65,54 | 70,77 | 77,23 | 48,46 | 33,98 | 75,23 |

### 7.4.36 Cosine distance and complete



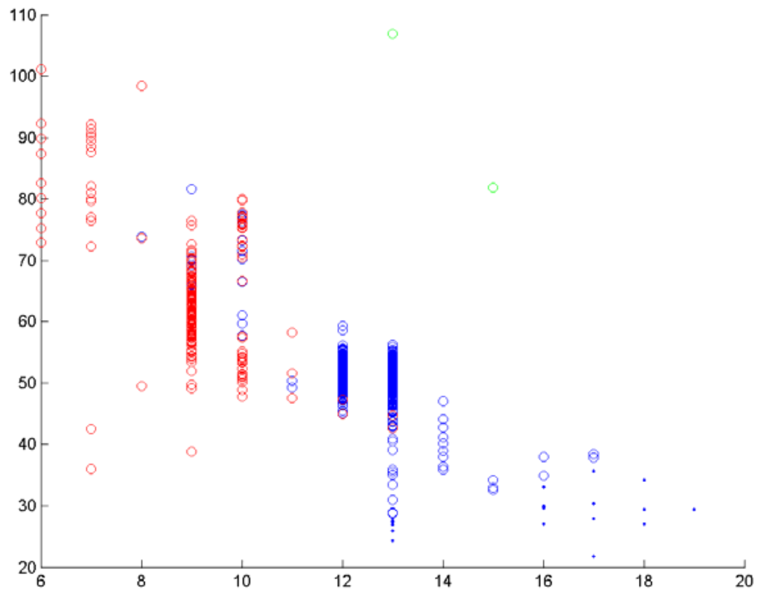
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 81,87 | 82,86 | 81,29 | 70,24 | 75,50 | 65,71 | 69,52 | 82,59 | 54,06 | 67,96 | 77,09 |

### 7.4.37 Cosine distance and single linkage



| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 21,79 | 81,00 | 81,29 | 70,66 | 73,27 | 65,71 | 70,77 | 83,04 | 49,86 | 33,01 | 75,00 |

### 7.4.38 Cosine distance and weighted linkage

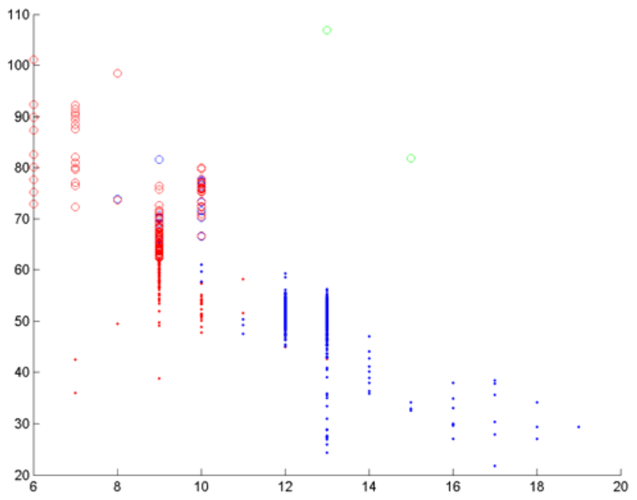


| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 83,61 | 57,98 | 81,29 | 77,01 | 72,40 | 65,71 | 70,23 | 82,59 | 48,46 | 66,83 | 73,02 |

### *K-means clustering*

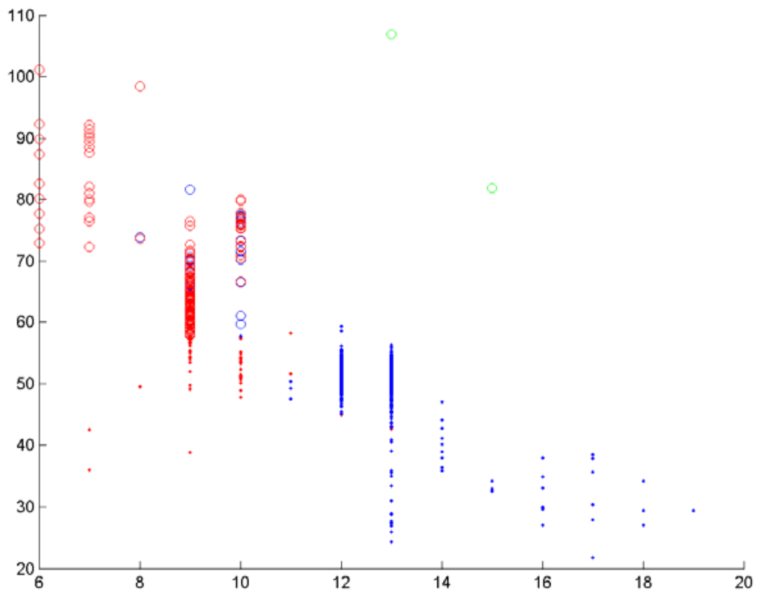
For the K-means clustering different distance metrics can be considered. The cases which were studied are three: squared Euclidean distance, Manhattan distance and cosine distance

### *7.4.39 Squared Euclidean distance*



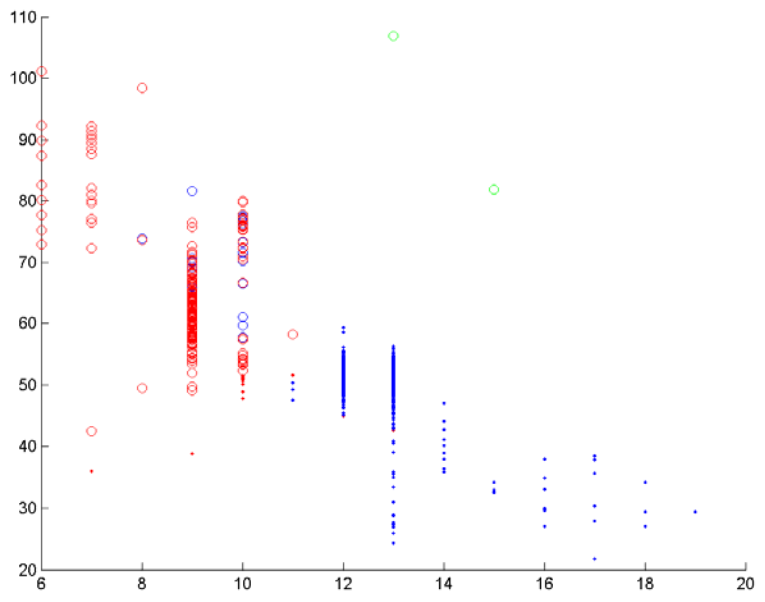
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 82,97 | 81,68 | 81,29 | 70,66 | 75,25 | 65,54 | 70,77 | 77,23 | 48,46 | 33,98 | 75,23 |

### *7.4.40 Manhattan distance*



| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 81,87 | 82,86 | 81,29 | 70,24 | 75,50 | 65,71 | 69,52 | 82,59 | 54,06 | 67,96 | 77,09 |

### 7.4.41 Cosine distance

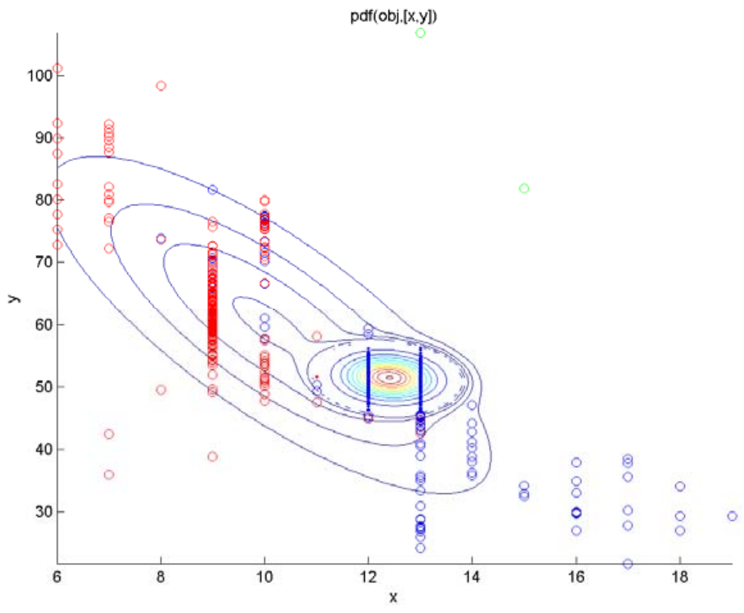


| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 21,79 | 81,00 | 81,29 | 70,66 | 73,27 | 65,71 | 70,77 | 83,04 | 49,86 | 33,01 | 75,00 |

### Multivariate Gaussian mixture models clustering

Regarding multivariate Gaussian mixture models clustering the options which were compared are two, the  case where the covariance matrix is diagonal and the case where it is a full matrix (not diagonal).

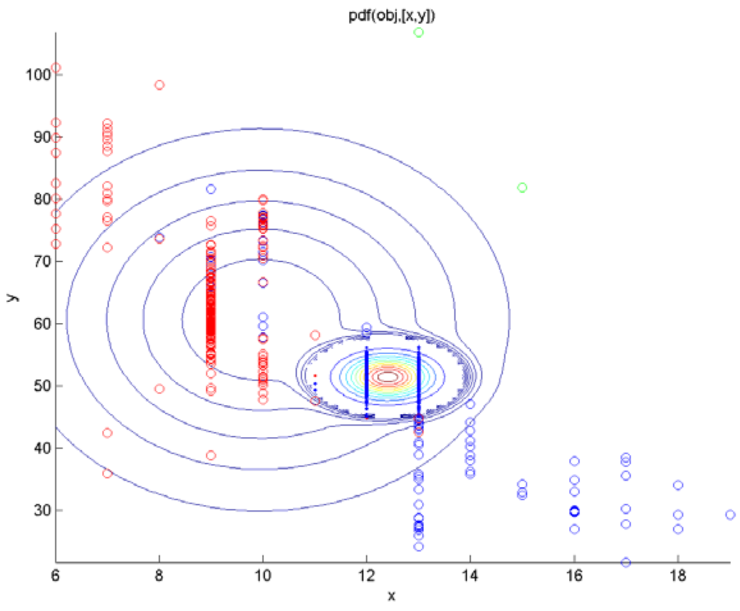
### 7.4.42 Full Covariance matrix



| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|

|  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| 82,97 | 81,68 | 81,29 | 70,66 | 75,25 | 65,54 | 70,77 | 77,23 | 48,46 | 33,98 | 75,23 |

### *7.4.43 Diagonal Covariance matrix*

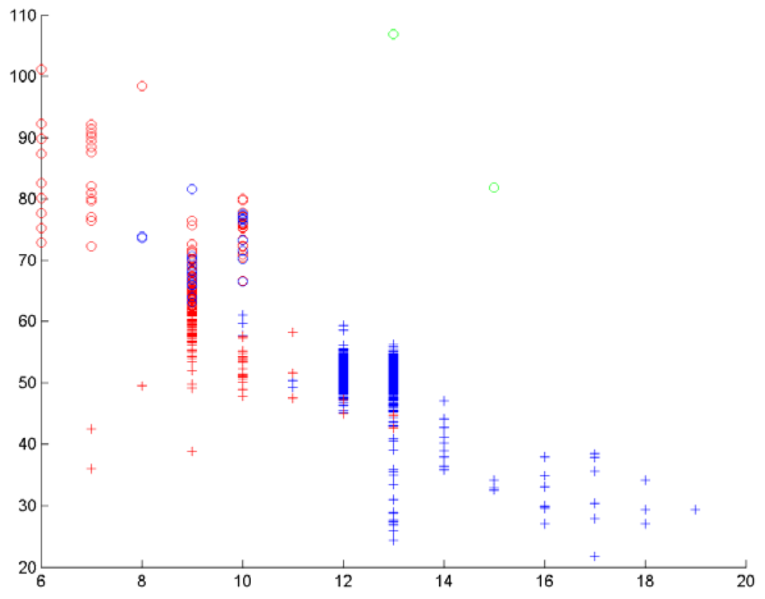


| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 81,87 | 82,86 | 81,29 | 70,24 | 75,50 | 65,71 | 69,52 | 82,59 | 54,06 | 67,96 | 77,09 |

### *Fuzzy k means*

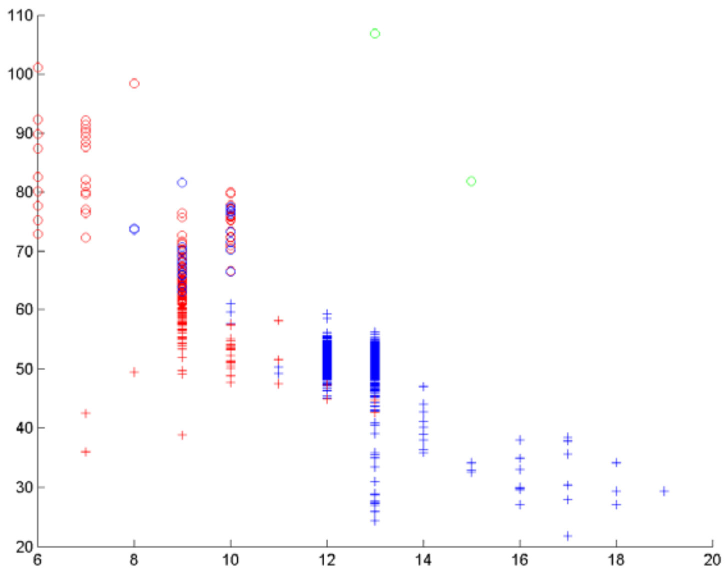
Fuzzy K-means clustering allows some parametric control with the fuzziness coefficient, so different values will be experimented: 2, 3, 5, 10, 100 and 250.

### *7.4.44 Fuzziness coefficient = 2*



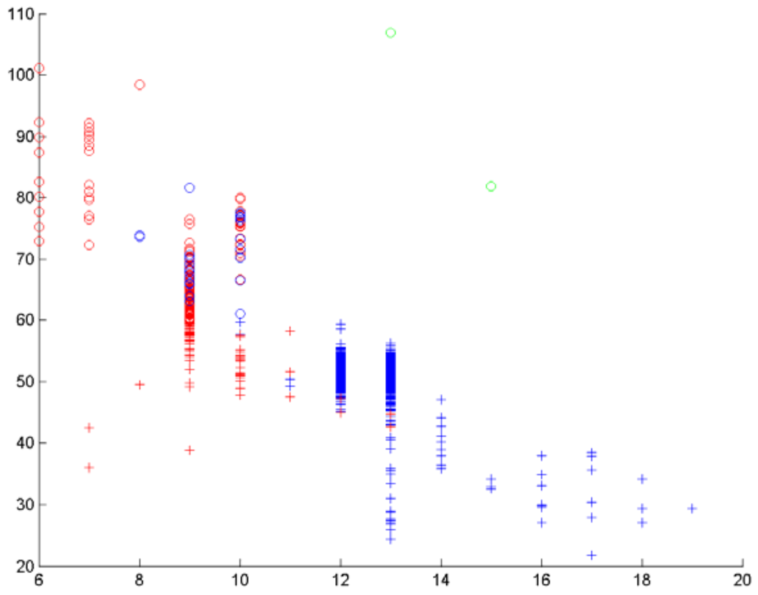
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 82,97 | 81,68 | 81,29 | 70,66 | 75,25 | 65,54 | 70,77 | 77,23 | 48,46 | 33,98 | 75,23 |

### 7.4.45 Fuzziness coefficient = 3



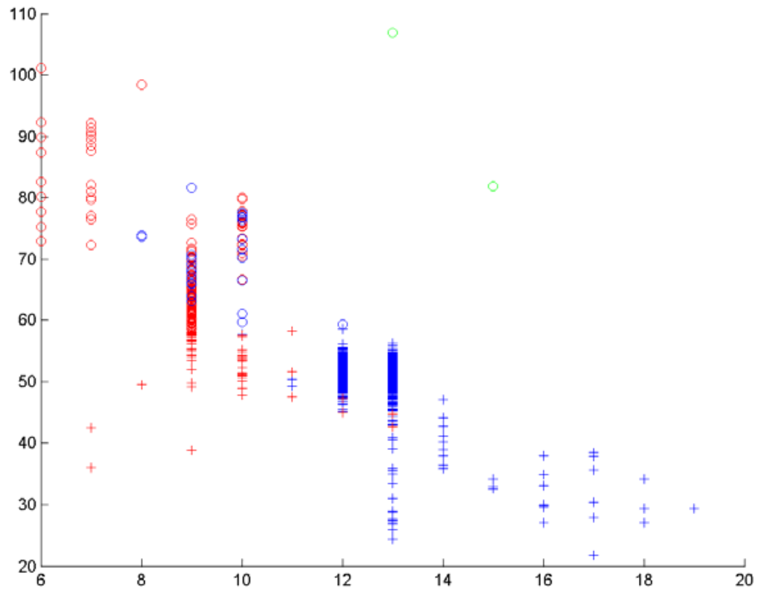
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 81,87 | 82,86 | 81,29 | 70,24 | 75,50 | 65,71 | 69,52 | 82,59 | 54,06 | 67,96 | 77,09 |

### 7.4.46 Fuzziness coefficient = 5



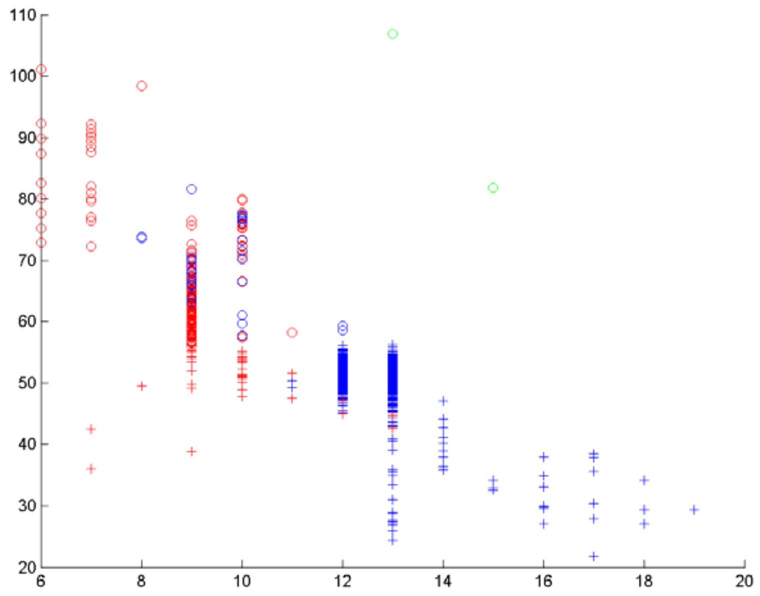
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 21,79 | 81,00 | 81,29 | 70,66 | 73,27 | 65,71 | 70,77 | 83,04 | 49,86 | 33,01 | 75,00 |

### *7.4.47 Fuzziness coefficient = 10*



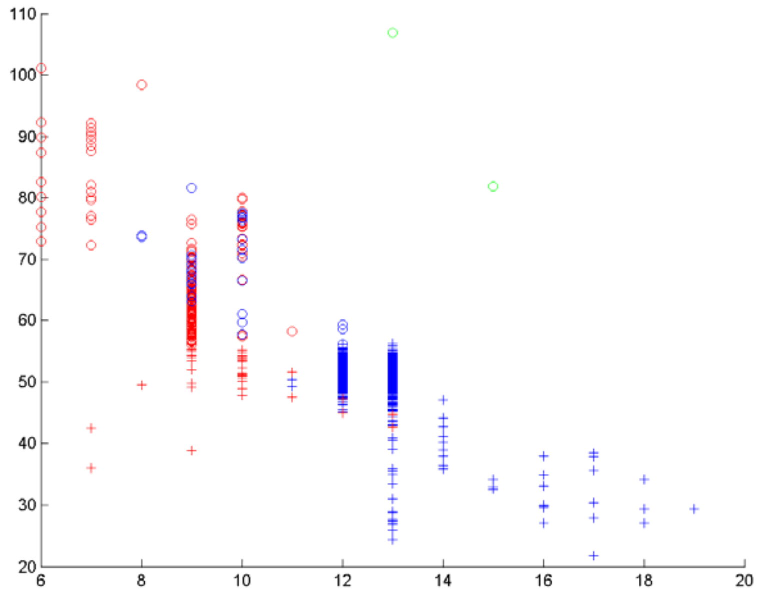
| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 83,61 | 57,98 | 81,29 | 77,01 | 72,40 | 65,71 | 70,23 | 82,59 | 48,46 | 66,83 | 73,02 |

### *7.4.48 Fuzziness coefficient = 100*



| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 82,97 | 81,68 | 81,29 | 70,66 | 75,25 | 65,54 | 70,77 | 77,23 | 48,46 | 33,98 | 75,23 |

### 7.4.49 Fuzziness coefficient = 250



| doc 1 | doc 2 | doc 3 | doc 4 | doc 5 | doc 6 | doc 7 | doc 8 | doc 9 | doc 10 | doc 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 81,87 | 82,86 | 81,29 | 70,24 | 75,50 | 65,71 | 69,52 | 82,59 | 54,06 | 67,96 | 77,09 |

## 7.5 Success rates for all datasets, techniques and their variants. Summary table

In the following table some abbreviations were committed due to the lack of space. The meanings are:

- av = average
- co = complete
- si = single
- we = weighted
- Wa = Ward
- me = median
- ce = centroid
- sq = Squared Euclidean
- Ma = Manhattan
- cos = cosine
- fu = full
- di = diagonal

The colored cells of following table represent the better results, being the lightest the worst and the darkest the best.

| Clust. | Dist | Link | d1 | d2 | d3 | d4 | d5 | d6 | d7 | d8 | d9 | d10 | d11 | Mean | Std Dev |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hierarchical agglomerative | Euclidean | av | 83,0 | 81,7 | 81,3 | 70,7 | 75,2 | 65,5 | 70,8 | 77,2 | 48,5 | 34,0 | 75,2 | 69,4 | 15,2 |
| | | co | 81,9 | 82,9 | 81,3 | 70,2 | 75,5 | 65,7 | 69,5 | 82,6 | 54,1 | 68,0 | 77,1 | 73,5 | 9,0 |
| | | si | 21,8 | 81,0 | 81,3 | 70,7 | 73,3 | 65,7 | 70,8 | 83,0 | 49,9 | 33,0 | 75,0 | 64,1 | 20,5 |
| | | we | 83,6 | 58,0 | 81,3 | 77,0 | 72,4 | 65,7 | 70,2 | 82,6 | 48,5 | 66,8 | 73,0 | 70,8 | 10,8 |
| | | Wa | 83,4 | 84,1 | 81,3 | 79,5 | 74,1 | 69,8 | 69,5 | 77,2 | 53,8 | 34,0 | 77,1 | 71,3 | 15,0 |
| | | me | 82,7 | 84,0 | 81,3 | 69,8 | 74,9 | 66,2 | 71,8 | 82,6 | 48,5 | 61,5 | 60,6 | 71,3 | 11,4 |
| | | ce | 83,0 | 81,7 | 81,3 | 70,7 | 75,2 | 65,5 | 70,8 | 77,2 | 48,5 | 34,0 | 75,2 | 69,4 | 15,2 |
| | sq. Euclidean | av | 21,8 | 18,5 | 17,4 | 28,8 | 76,4 | 65,7 | 28,7 | 13,4 | 48,7 | 33,0 | 76,2 | 39,0 | 23,8 |
| | | co | 87,5 | 84,9 | 81,3 | 70,0 | 74,9 | 65,7 | 77,7 | 87,9 | 48,7 | 33,0 | 76,7 | 71,7 | 17,1 |
| | | si | 21,8 | 18,5 | 17,4 | 28,8 | 25,0 | 65,7 | 28,7 | 13,4 | 49,9 | 33,0 | 24,3 | 29,7 | 15,4 |
| | | we | 82,4 | 18,5 | 17,4 | 89,0 | 75,0 | 65,7 | 28,7 | 77,2 | 49,9 | 33,0 | 76,2 | 55,7 | 27,0 |
| | | Wa | 96,7 | 95,3 | 91,9 | 80,0 | 89,9 | 69,9 | 92,7 | 81,7 | 54,1 | 69,9 | 94,4 | 83,3 | 13,7 |
| | | me | 82,4 | 18,5 | 17,4 | 69,1 | 74,9 | 65,7 | 70,8 | 13,4 | 49,9 | 33,0 | 76,2 | 51,9 | 26,5 |
| | | ce | 81,8 | 18,5 | 17,4 | 28,8 | 75,0 | 65,7 | 28,7 | 13,4 | 48,7 | 33,0 | 76,7 | 44,3 | 26,1 |
| | Manhattan | av | 82,4 | 81,2 | 81,3 | 70,7 | 74,9 | 65,5 | 70,8 | 78,6 | 48,5 | 34,0 | 75,2 | 69,4 | 15,2 |
| | | co | 93,3 | 81,5 | 81,3 | 69,8 | 75,5 | 66,4 | 70,8 | 77,2 | 48,5 | 33,8 | 74,9 | 70,3 | 16,4 |
| | | si | 21,8 | 81,0 | 17,4 | 70,7 | 73,3 | 65,7 | 70,8 | 88,8 | 49,9 | 33,0 | 75,0 | 58,8 | 24,5 |
| | | we | 93,3 | 81,2 | 81,3 | 70,7 | 74,1 | 65,7 | 70,2 | 80,8 | 48,5 | 33,0 | 75,2 | 70,4 | 16,8 |
| | Minkowski exp 0,5 | av | 81,2 | 81,0 | 81,3 | 79,3 | 74,9 | 65,5 | 70,8 | 77,2 | 48,5 | 33,8 | 74,9 | 69,9 | 15,3 |
| | | co | 94,4 | 84,4 | 81,3 | 74,5 | 81,4 | 66,4 | 70,2 | 77,2 | 51,3 | 33,8 | 76,0 | 71,9 | 16,7 |
| | | si | 21,8 | 81,0 | 80,3 | 70,2 | 73,3 | 65,7 | 70,8 | 83,0 | 49,9 | 33,0 | 75,2 | 64,0 | 20,4 |
| | | we | 30,9 | 83,6 | 81,3 | 72,5 | 75,4 | 65,7 | 70,2 | 83,0 | 51,3 | 33,8 | 77,2 | 65,9 | 19,0 |
| | Minkowski exp 3 | av | 21,8 | 18,5 | 17,4 | 28,8 | 76,4 | 65,7 | 28,7 | 13,4 | 48,7 | 33,0 | 76,2 | 39,0 | 23,8 |
| | | co | 87,5 | 84,9 | 81,3 | 70,0 | 74,9 | 65,7 | 77,7 | 87,9 | 48,7 | 33,0 | 76,7 | 71,7 | 17,1 |
| | | si | 21,8 | 18,5 | 17,4 | 28,8 | 25,0 | 65,7 | 28,7 | 13,4 | 49,9 | 33,0 | 24,3 | 29,7 | 15,4 |
| | | we | 82,4 | 18,5 | 17,4 | 89,0 | 75,0 | 65,7 | 28,7 | 77,2 | 49,9 | 33,0 | 76,2 | 55,7 | 27,0 |
| | Chebychev | av | 83,0 | 81,7 | 81,3 | 70,7 | 75,2 | 65,5 | 70,6 | 77,2 | 48,5 | 34,0 | 75,0 | 69,3 | 15,2 |
| | | co | 87,5 | 83,6 | 81,3 | 70,7 | 74,3 | 65,7 | 70,6 | 82,6 | 48,5 | 33,0 | 76,2 | 70,4 | 16,4 |
| | | si | 21,8 | 81,0 | 81,3 | 70,7 | 73,3 | 65,7 | 70,8 | 83,9 | 49,9 | 33,0 | 75,0 | 64,2 | 20,6 |
| | | we | 83,6 | 84,0 | 81,3 | 69,8 | 74,3 | 66,2 | 70,2 | 82,6 | 48,5 | 66,8 | 53,4 | 71,0 | 12,0 |
| | Mahalanobis | av | 21,8 | 18,5 | 18,4 | 28,8 | 73,1 | 65,7 | 28,7 | 13,4 | 49,9 | 33,0 | 74,3 | 38,7 | 23,0 |
| | | co | 21,8 | 81,9 | 18,4 | 70,2 | 74,3 | 65,2 | 68,8 | 19,6 | 49,9 | 33,0 | 76,4 | 52,7 | 24,9 |
| | | si | 21,8 | 18,5 | 17,4 | 28,8 | 25,0 | 65,7 | 28,7 | 13,4 | 49,9 | 33,0 | 24,3 | 29,7 | 15,4 |
| | | we | 21,8 | 18,4 | 17,4 | 28,8 | 67,0 | 65,7 | 28,7 | 83,0 | 49,9 | 33,0 | 75,8 | 44,5 | 24,5 |
| | cosine | av | 25,5 | 18,4 | 83,2 | 83,1 | 27,0 | 70,3 | 29,1 | 76,8 | 55,5 | 33,0 | 24,4 | 47,8 | 26,1 |
| | | co | 24,6 | 18,4 | 91,6 | 78,8 | 43,1 | 80,9 | 73,8 | 75,9 | 69,7 | 33,3 | 24,7 | 55,9 | 27,2 |
| | | si | 21,9 | 18,5 | 17,7 | 28,9 | 25,1 | 65,7 | 70,8 | 14,7 | 50,1 | 32,8 | 24,4 | 33,7 | 19,6 |
| | | we | 23,7 | 60,0 | 28,7 | 83,1 | 76,2 | 73,6 | 70,9 | 74,6 | 66,4 | 32,8 | 81,5 | 61,1 | 22,0 |
| K-means | | sq | 87,5 | 83,0 | 85,2 | 78,7 | 75,2 | 69,8 | 69,5 | 77,2 | 51,8 | 66,8 | 77,1 | 74,7 | 10,0 |
| | | Ma | 92,2 | 54,8 | 89,0 | 79,1 | 63,5 | 57,9 | 45,8 | 67,9 | 54,3 | 70,2 | 63,0 | 67,1 | 14,7 |
| | | cos | 95,9 | 93,4 | 91,6 | 83,6 | 84,3 | 81,6 | 82,2 | 75,9 | 69,2 | 78,5 | 84,7 | 83,7 | 7,8 |
| Gmm | | fu | 91,1 | 95,0 | 81,3 | 87,0 | 85,0 | 87,7 | 78,6 | 80,8 | 56,0 | 78,5 | 94,4 | 83,2 | 10,8 |
| | | di | 91,8 | 95,0 | 82,9 | 86,2 | 85,9 | 86,7 | 76,6 | 80,8 | 55,5 | 77,3 | 78,3 | 81,5 | 10,4 |
| fuzzy K-means | | 2 | 88,0 | 83,2 | 82,9 | 78,7 | 73,3 | 69,6 | 69,3 | 77,2 | 52,1 | 66,8 | 77,1 | 74,4 | 9,9 |
| | | 3 | 89,0 | 83,9 | 83,2 | 78,7 | 63,5 | 69,9 | 68,3 | 77,2 | 53,5 | 68,6 | 77,1 | 73,9 | 10,3 |
| | | 5 | 90,4 | 85,2 | 85,2 | 78,7 | 61,9 | 69,9 | 51,5 | 76,8 | 53,8 | 68,1 | 68,5 | 71,8 | 12,8 |
| | | 10 | 91,4 | 80,8 | 87,4 | 78,4 | 61,3 | 68,6 | 49,9 | 76,3 | 51,8 | 68,3 | 66,6 | 71,0 | 13,4 |
| | | 100 | 93,4 | 63,8 | 88,7 | 75,2 | 58,9 | 63,7 | 51,9 | 61,6 | 51,3 | 59,7 | 64,3 | 66,6 | 13,7 |
| | | 250 | 93,3 | 81,9 | 88,4 | 78,6 | 58,7 | 66,6 | 47,6 | 65,2 | 56,6 | 66,8 | 66,2 | 70,0 | 14,0 |

There are several combinations that have unsatisfactory results, but most of them are near the 70% of success rate, which is not bad. It would mean that 7 out of 10 elements are correctly recognized.

Only hierarchical agglomerative clustering with squared Euclidean distance and ward linkage, K-means and cosine distance and multivariate Gaussian mixture models reach an success rate higher than 80%. It would mean that more than 8 out of 10 elements are correctly recognized.

| Clust. | Dist | Link | d1 | d2 | d3 | d4 | d5 | d6 | d7 | d8 | d9 | d10 | d11 | Mean | Std Dev |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hier. Agglo. | sq. Eu | wa | 96,7 | 95,3 | 91,9 | 80,0 | 89,9 | 69,9 | 92,7 | 81,7 | 54,1 | 69,9 | 94,4 | 83,3 | 13,7 |
| K-means | | co | 95,9 | 93,4 | 91,6 | 83,6 | 84,3 | 81,6 | 82,2 | 75,9 | 69,2 | 78,5 | 84,7 | 83,7 | 7,8 |
| Gmm | | fu | 91,1 | 95,0 | 81,3 | 87,0 | 85,0 | 87,7 | 78,6 | 80,8 | 56,0 | 78,5 | 94,4 | 83,2 | 10,8 |

*Figure 56 – Clustering techniques with better results for all the datasets.*

K-means clustering and cosine distance has the best mean success rate considering all the datasets and, furthermore, it seems to be the robustest one, as it indicates its standard deviation. It also has the best success rate in the document 9 where all the techniques decrease their success rates drastically.

If we want to remove the dataset documents that have an abnormally low success rate (documents 6, 9 and 10) the success rate can reach higher values than 90%, which means that more than 9 out of 10 elements are correctly recognized. Then, the hierarchical agglomerative clustering with squared Euclidean distance and Ward linkage would clearly have the best success rate and they would be the robustest.

| Clust. | Dist | Link | 1 | 2 | 3 | 4 | 5 | 7 | 8 | 11 | mean | std. dev. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hier. Agglo. | sq. Eu | wa | 96,7 | 95,3 | 91,9 | 80,0 | 89,9 | 92,7 | 81,7 | 94,4 | 90,3 | 6,2 |
| K-means | | co | 95,9 | 93,4 | 91,6 | 83,6 | 84,3 | 82,2 | 75,9 | 84,7 | 86,4 | 6,7 |
| Gmm | | fu | 91,1 | 95,0 | 81,3 | 87,0 | 85,0 | 78,6 | 80,8 | 94,4 | 86,7 | 6,3 |

*Figure 57 – Considering only the datasets where the algorithms perform the best results.*

# 8 Conclusions

The main conclusion that can be drawn from this work is that the proposed challenge can be achieved. The present work covers all the initial points successfully. Furthermore, the success rates are quite high (average near 90%). Previously to the beginning of the project, and due to fact that the subject was totally innovative, the question about a possible solution to visual drawbacks derived from reading in small devices didn't have an answer. Now there is an open door to an unexplored field which seems to have an important place in the future daily life.

The project tackled a determined type of text documents but the approach is open to be improved in order to spread the range of document types which are able to be processed by the proposed methodology.

Although a document text can have innumerable possibilities of format, fonts and structure, there won't be an algorithm that could always work well. Anyway, the proposed solution offers robustness to failures. Three of the forty nine algorithm variants experimented, belonging to four clustering families, showed excellent success rates and quite acceptable robustness. The best algorithms were K-means using cosine distance, hierarchical agglomerative clustering using squared Euclidean distance and Ward linkage, and multivariate Gaussian mixture models using a full covariance matrix option.

This successful result encourages future researches on this topic which could finally lead to a practical solution, materialized in a software solution.

# 9 Future works

Regarding the methodology proposed, there is one step further it can go. It is possible to build an ensemble with the algorithms studied. It would improve the performance of the solution.

Basically, the future works can be divided into researches related to a real software solution, researches related to the application and possible benefits in daily life and researches related to artificial intelligence and computer vision.

## 9.1 A real software solution.

Luckily, the work on this topic has not finished yet. We still have a long way to walk until a final application could be integrated in daily life as other technological advances had.

After achieving the initial challenge which this research proposes, the classification of the information extracted from the document text can be done at real time with a high successful rate. Thus, we could start the next advances from that point onwards.

The following step would be developing a user interface which would ask the user to customize some of the most representative instances of every class. Then the set of preferences would be extrapolated to the rest of the instances. Also, the application could keep a record of the most representative decisions taken by the user, as the case-based reasoning algorithms would do it. In that manner, the algorithm could start to take its own decisions in repetitive cases or notify the user that some customizing decision goes against decisions taken previously.

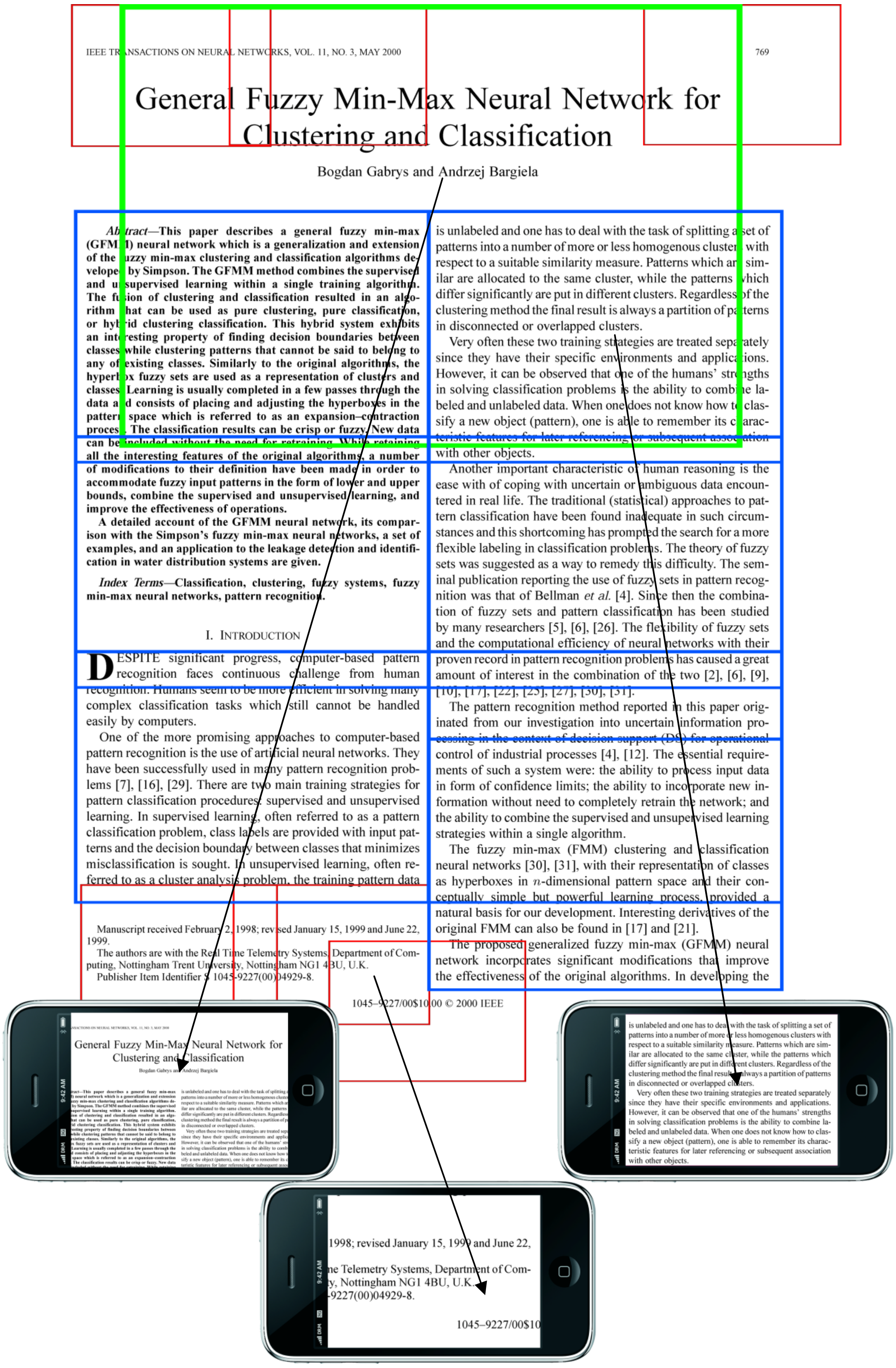
Once this interaction with the user is covered, the decisions could be applied to the rest of the instances with fuzzy logic techniques. It would propitiate that the extrapolation of the decisions would become flexible and auto-adapt to every instance.

Regarding soft computing techniques, the next step would be elaborating an optimal graphic sequence strategy based on the previous customization and adaptation. It means that the application should create a sequence of scrollings and zoomings that would move along the document optimizing the visualization and, at the same time, minimizing the interaction by the user. Here, the optimization of different features at the same time seems to fit with multiobjective metaheuristics.

The last step would be the creation of a user interface where the optimal graphic sequence could be performed with some options which would let the user amend the possible cases classified wrongly, navigate through the document with the application assistance on/off, rectify the customization, etc.

In the best case, the user would only have to jump to the next scrolling and zooming position that the application chose by pressing a button ‘next’, ‘forward’ and ‘backward’, double clicking in different areas of the screen, etc. For example, double clicking on the right side of the screen would go forward and on left side would go backwards.

The popular *grab & drag* and *pich open & close* tools for interacting with touching screens could be present at the same time too.



IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 11, NO. 3, MAY 2000 769

General Fuzzy Min-Max Neural Network for Clustering and Classification

Bogdan Gabrys and Andrzej Bargiela

*Abstract*—This paper describes a general fuzzy min-max (GFMM) neural network which is a generalization and extension of the fuzzy min-max clustering and classification algorithms developed by Simpson. The GFMM method combines the supervised and unsupervised learning within a single training algorithm. The fusion of clustering and classification resulted in an algorithm that can be used as pure clustering, pure classification, or hybrid clustering classification. This hybrid system exhibits an interesting property of finding decision boundaries between classes while clustering patterns that cannot be said to belong to any of existing classes. Similarly to the original algorithms, the hyperbox fuzzy sets are used as a representation of clusters and classes. Learning is usually completed in a few passes through the data and consists of placing and adjusting the hyperboxes in the pattern space which is referred to as an expansion–contraction process. The classification results can be crisp or fuzzy. New data can be included without the need for retraining. While retaining all the interesting features of the original algorithms, a number of modifications to their definition have been made in order to accommodate fuzzy input patterns in the form of lower and upper bounds, combine the supervised and unsupervised learning, and improve the effectiveness of operations.

A detailed account of the GFMM neural network, its comparison with the Simpson's fuzzy min-max neural networks, a set of examples, and an application to the leakage detection and identification in water distribution systems are given.

*Index Terms*—Classification, clustering, fuzzy systems, fuzzy min-max neural networks, pattern recognition.

I. INTRODUCTION

DESPITE significant progress, computer-based pattern recognition faces continuous challenge from human recognition. Humans seem to be more efficient in solving many complex classification tasks which still cannot be handled easily by computers.

One of the more promising approaches to computer-based pattern recognition is the use of artificial neural networks. They have been successfully used in many pattern recognition problems [7], [16], [29]. There are two main training strategies for pattern classification procedures: supervised and unsupervised learning. In supervised learning, often referred to as a pattern classification problem, class labels are provided with input patterns and the decision boundary between classes that minimizes misclassification is sought. In unsupervised learning, often referred to as a cluster analysis problem, the training pattern data is unlabeled and one has to deal with the task of splitting a set of patterns into a number of more or less homogenous clusters with respect to a suitable similarity measure. Patterns which are similar are allocated to the same cluster, while the patterns which differ significantly are put in different clusters. Regardless of the clustering method the final result is always a partition of patterns in disconnected or overlapped clusters.

Very often these two training strategies are treated separately since they have their specific environments and applications. However, it can be observed that one of the humans' strengths in solving classification problems is the ability to combine labeled and unlabeled data. When one does not know how to classify a new object (pattern), one is able to remember its characteristic features for later referencing or subsequent association with other objects.

Another important characteristic of human reasoning is the ease with of coping with uncertain or ambiguous data encountered in real life. The traditional (statistical) approaches to pattern classification have been found inadequate in such circumstances and this shortcoming has prompted the search for a more flexible labeling in classification problems. The theory of fuzzy sets was suggested as a way to remedy this difficulty. The seminal publication reporting the use of fuzzy sets in pattern recognition was that of Bellman *et al.* [4]. Since then the combination of fuzzy sets and pattern classification has been studied by many researchers [5], [6], [26]. The flexibility of fuzzy sets and the computational efficiency of neural networks with their proven record in pattern recognition problems has caused a great amount of interest in the combination of the two [2], [6], [9], [10], [17], [22], [25], [27], [30], [31].

The pattern recognition method reported in this paper originated from our investigation into uncertain information processing in the context of decision support (DS) for operational control of industrial processes [4], [12]. The essential requirements of such a system were: the ability to process input data in form of confidence limits; the ability to incorporate new information without need to completely retrain the network; and the ability to combine the supervised and unsupervised learning strategies within a single algorithm.

The fuzzy min-max (FMM) clustering and classification neural networks [30], [31], with their representation of classes as hyperboxes in $n$-dimensional pattern space and their conceptually simple but powerful learning process, provided a natural basis for our development. Interesting derivatives of the original FMM can also be found in [17] and [21].

The proposed generalized fuzzy min-max (GFMM) neural network incorporates significant modifications that improve the effectiveness of the original algorithms. In developing the

Manuscript received February 2, 1998; revised January 15, 1999 and June 22, 1999.

The authors are with the Real Time Telemetry Systems, Department of Computing, Nottingham Trent University, Nottingham NG1 4BU, U.K.

Publisher Item Identifier S 1045-9227(00)04929-8.

1045–9227/00$10.00 © 2000 IEEE

Figure 58 – Graphical simulation of the scrollings and zooming created by the proposed application

## 9.2 Applications and benefits in daily life.

As it was mentioned in the state of art, there are several researches about the amount of scrolling needed to read a document and its influence on the readability and comprehension of the reading. Also, there are researches on the influence of the screen sizes and the font size, and hence, on the zooming needed. It should be remembered that all the papers can be summarized in the idea that the user reduces the reading comprehension when parallel mental processes to read are active. The main cases are zooming and scrolling derived of the use of small screens and very small or very big fonts.

An example of the benefits of this kind of software tools in society would be the aid that they could suppose to disabled persons. Tetraplegic individuals or people with reduced mobility in their upper limbs or hands, like the well-known case of Stephen Hawkins, would extremely appreciate a software tool which could minimize their interaction with control-related hardware devices.

Another possible benefit to the society would be the possibility to detect issues related to sight problems in children in very early ages. The study of the amount of interactions that children would have with the software could stand out some suspicious patterns related to any type of disease or deficiency. Those issues, commonly hidden or really difficult to detect for adults, can be discovered automatically. It would be useful for grown children, at the age when the reading learning starts, to help to detect dyslexia or attention deficit hyperactivity disorder (ADHD) [48]. It also can be used to periodic adult revisions or rehab processes. It is also remarkable the great help that it would suppose in the detection of early Alzheimer cases in mature people.

## 9.3 Artificial intelligence and computer vision.

The research could be transferred to other subjects in the field of artificial intelligence. The development of this technology could open a possibility to processing intelligently captured text in every possible media, i.e. with an image recorder (camera, video recorder or similar). It would allow artificial devices to have the capability of reading as humans do. This kind of development would open the door to further steps in computer vision and artificial intelligence. If we consider the achievements in soft computing techniques related with fuzzy logic and computing with perceptions, and the interaction humans-robots [15] [47] and the natural language, the possibilities are immense.

# 10 References

[1] List of displays by pixels density. - Wikipedia
http://en.wikipedia.org/wiki/List_of_displays_by_pixel_density

[2] Interacting with Big Interfaces on Small Screens: a Comparison of Fisheye, Zoom, and Panning Techniques.
*Carl Gutwin and Chris Fedak*

[3] RotoView.
*www.rotoview.com*

[4] Mac finger gestures. – Apple
*http://www.apple.com/macosx/whats-new/gestures.html*

[5] Leap Motion.
*www.leapmotion.com*

[6] The effect of display size and text splitting on reading lengthy text from the screen.
*A. Dillon, J. Richardson and C. McKnight,*

[7] Readability of text scrolled on visual display terminals as a function of window size.
*R.L. Duchnicky and P.A. Kolers,*

[8] The Effect of Age and Font Size on Reading Text on Handheld Computers.
*Iain Darroch, Joy Goodman, Stephen Brewster and Phil Gray*

[9] Presenting information on the small-screen interface: effects of table formatting.
*Kim, L., Albers, M.J.*

[10] Document page segmentation using neuro-fuzzy approach.
*Laura Caponetti, Ciro Castiello, Przemysław Go´recki*

[11] A knowledge-based system for extracting text-lines from mixed and overlapping text/graphics compound document images.
*Yen-Lin Chen a, Zeng-Wei Hong b, Cheng-Hung Chuang*

[12] Page segmentation of Chinese newspapers.
*Jie Xi ∗, Jianming Hu, Lide Wu*

[13] Page Segmentation Using the Description of the Background.
*Apostolos Antonacopoulos*

[14] Comic Image Decomposition for Reading Comics on Cellular Phones.
*Masashi YAMADA, Rahmat BUDIARTO, Mamoru ENDO, Shinya MIYAZAKI*

[15] Human-Robot Interaction through Spoken Language Dialogue.
*L. Seabra Lopes and A. Teixeira*

[16] Ultra High Definition Television – Wikipedia.
*http://en.wikipedia.org/wiki/Ultra_High_Definition_Television*

[17] Rapid Serial Visual Presentation Techniques for Consumer Digital Video Devices.
*Kent Wittenburg, Clifton Forlines, Tom Lanning and Taizo Miyachi*

[18] General Fuzzy Min-Max Neural Network for Clustering and Classification.
*Bogdan Gabrys and Andrzej Bargiela*

[19] Colour text segmentation in web images based on human perception.
*D. Karatzas a, A. Antonacopoulos b*

[20] Method for Real Time Text Extraction of Digital Manga Comic.
*Kohei Arai, Herman Tolle*

[21] Identification of Text-Only Areas in Mixed-Type Documents.
*C. Strouthopoulos, N. Papamarkos, C. Chamzas*

[22] Page segmentation and identification for intelligent signal processing.
*Kuo-Chin Fan, Liang-Shen Wang, Yuan-Kai Wang*

[23] Optimizing the reading of electronic text using rapid serial visual presentation.
*Monica S. Castelhano and Paul Muter*

[24] Text quality metrics for visual display units: I. Methodological aspects.
*Jacques A.J. Roufs, Martin C. Boschman*

[25] Reading text presented on a small display.
*James F. Juola, Alp Tiritoglu' and John Pleunis*

[26] Using large tables on small display devices.
*Carolyn Watters, Jack Duffy, Kathryn Duffy*

[27] Segmentation of page images having artifacts of photocopying and scanning.
*L. Cinquea; S. Levialdia, L. Lombardib, S. Tanimotoc*

[28] Color text extraction with selective metric-based clustering .
*Celine Mancas-Thillou, Bernard Goss*

[29] Cubic convolution interpolation for digital image processing.
*R. Key*

[30] Nearest Neighbor Value Interpolation.
*Rukundo Olivier, Cao Hanqiang*

[31] Image Analysis and Mathematical Morphology.
*Jean Serra*

[32] Minkowski Sums of Monotone and General Simple Polygons.
*Oks , Eduard; Sharir, Micha*

[33] Morphological Image Analysis: Principles and Applications.
*Springer-Verlag,*

[34] Connected Component Labeling.
*R. Fisher, S. Perkins, A. Walker and E. Wolfart*

[35] Hierarchical Clustering Schemes, Psychometrika
*S. C. Johnson*

[36] ^ "The DISTANCE Procedure: Proximity Measures".
*SAS Institute.*

[37] Hierarchical Grouping to Optimize an Objective Function,
*Ward, J. H., Jr.*

[38] Cluster Analysis
*Brian S. Everitt, Sabine Landau, and Morven Leese*

[39] Complexities of Hierarchic Clustering Algorithms: the state of the art Computational.
*Murtagh F,*

[40] Some Methods for classification and Analysis of Multivariate Observations.
*J. B. MacQueen*

[41] Data Mining - Clustering by Mixture Models.
*Jia Li*

[42] Gaussian Mixture Models_
*Douglas Reynolds*

[43] Maximum Likelihood from Incomplete Data via theEM algorithm.
*A. P. Dempster, N.M. Laird, and D.B. Rubin*

[44] The EM algorithm, The Elements of Statistical Learning.
*Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome*

[45] A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters.
*J. C. Dunn*

[46] Pattern Recognition with Fuzzy Objective Function Algoritms.
*J. C. Bezdek*

[47] Multi-robot Cooperation for Human-Robot Communication
*Takayuki Kanda1, Tetsuo Ono, Michita Imai, and Kenji Mase1*

[48] Attention deficit hyperactivity disorder - wikipedia
*http://en.wikipedia.org/wiki/Attention_deficit_hyperactivity_disorder*