Spatial Fuzzy Clustering with Simultaneous Estimation of Markov Random Field Parameters and Class

Lledó Esquerra Ortells

e-mail: lledoe@gmail.com

Medical Imaging Research Center, Illinois Institute of Technology

Telecom BCN, Polytechnic University of Catalonia

Advisor: Prof. Dr. Miles N. Wernick

January 22, 2011

Abstract

Unsupervised Fuzzy C-Means (FCM) clustering technique has been widely used in image segmentation. However, conventional FCM algorithm has an intrinsic limitation: no spatial information is taken into account. This causes the FCM algorithm to work only on well-defined images with low level of noise. In this document a improvement to fuzzy clustering is described. A prior spatial constraint is introduced into FCM algorithm through Markov random field theory and its equivalent Gibbs random field theory, in which the spatial information is encoded through mutual influences of neighboring sites.

This unsupervised fuzzy Bayesian image segmentation method using fuzzy Markov random fields (FRMFs), which provides an improved segmentation results when compared to the 'hard' MRF method, has two groups of parameters to be estimated: the MRF parameters and class parameters for each pixel in the image. Typically, these two parameters are treated separately, and estimated in an alternating fashion. In this work, a method to estimate the parameters defining the Markovian distribution of the measured data while performing the data clustering simultaneously is developed. That is possible by defining estimates of the MRF parameters as functions of the class parameters resulting in a cost function that depends only on the class parameters of each pixel. The Conjugate Gradient method (CGM) is applied to search the optimizer of the resulting non-linear cost function.

We perform computer simulations on synthetic test images to demonstrate the efficacy and efficiency of the proposed method and also provide a comparison with some of the commonly used methods.

This work is only a little part of a more complex project based on predictive analytics. This big project works with geospatial data, therefore some knowledge about this type of data and how it has to be processed in order to be clustered by our method is also included.

Acknowledgements

I would like to acknowledge all the people who have helped me realize this Thesis. Specially, I want to thank my advisors, Prof. Dr. Miles N. Wernick and Rossella Blatt.

I would also thank the people of the International Relationship Office, Telecom BCN.

I do not forget my friends who had contributed with their patience throughout my studies.

Finally, but not less important, a special 'gràcies' to my mother and my brother, for their unconditional encouragement. And always remembering my father.

Contents

List of Figures

List of Tables

Chapter 1

Pattern Recognition

1.1 Introduction

Pattern recognition[53] has a long history, but before the 1960s it was mostly the output of theoretical research in the area of statistics. As with everything else, the advent of computers increased the demand for practical applications of pattern recognition, which in turn set new demands for further theoretical developments. As our society evolves from the industrial to its postindustrial phase, automation in industrial production and the need for information handling and retrieval are becoming increasingly important. This trend has pushed pattern recognition to the high edge of todays engineering applications and research. Pattern recognition is an integral part of most machine intelligence systems built for decision making.

Pattern recognition is the scientific discipline whose goal is the classification of objects into a number of categories or classes. Depending on the application, these objects can be images or signal waveforms or any type of measurements that need to be classified. We will refer to these objects using the generic term patterns or features. A pattern is the information that characterizes and differentiates the element to recognize. They should be extracted from the same data source, for example we cannot compare voice with image patterns (although we can fuse them to generate new patterns). In this work we will refer to them as vectors of features/characteristics or samples, or just pixels since they are going to be the objects which we will work with. A class refers to a group of vectors with similar characteristics.

Basically, the objective is to apply a classification algorithm to different classification problems in different fields or disciplines, as communications (symbol detection), machine vision (image processing), character (letter or number) recognition, computer-aided diagnosis, speech recognition, data mining and knowledge discovery in databases, and so one. There are a lot of areas in which pattern recognition is of importance. The foregoing are only some examples from a much larger number of possible applications. Of course, to achieve the final goals in all of these applications, pattern recognition is closely linked with other scientific disciplines, such as linguistics, computer graphics, machine vision, and database design.

Having aroused the readers curiosity about pattern recognition, we will next sketch the basic philosophy and methodological directions in which the various pattern recognition approaches have evolved and developed.

1.2 Features, Feature Vectors and Classifiers

Let us first simulate a simplified case 'mimicking' a medical image classification task. Fig. 1.1 shows two images, each having a distinct region inside it. The two regions are also themselves visually different. We could say that the region of the left results from a benign lesion, class A, and that of the right from a malignant one (cancer), class B. We will further assume that these are not the only patterns (images) that are available to us, but we have access to an image database with a number of patterns, some of which are known to originate from class A and some from class B. The first step is to identify the measurable

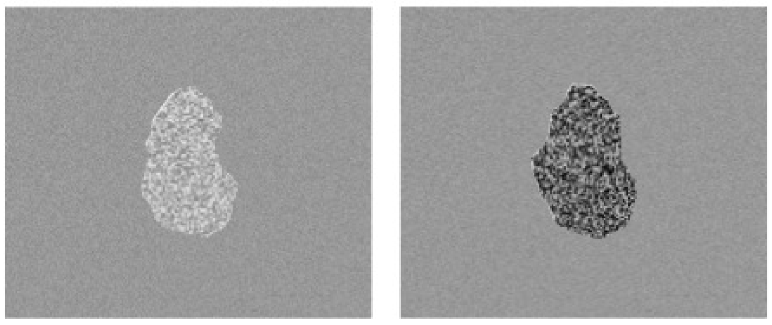

Figure 1.1: Examples of image regions corresponding to class A (left), and (b) class B (right)

quantities that make these two regions distinct from each other. Fig. 1.2[1] shows a plot of the mean value of the intensity in each region of interest versus the corresponding standard deviation around this mean. Each point corresponds to a different image from the available database (thus we can infer that in the database we have eight images from class A and eight from class B). It turns out that class A patterns tend to spread in a different area from class B patterns. The straight line seems to be a good candidate for separating the two classes.

Let us now assume that we are given a new image with a region in it and that we do not know to which class it belongs. It is reasonable to say that we measure the mean intensity and standard deviation in the region of interest and we plot the corresponding point. This is shown by the asterisk (*) in Fig. 1.2. Then it is sensible to assume that the unknown pattern is more likely to belong to class A than class B.

[1]This Figure corresponds to a 'Scatter Plot', a very useful way to represent our data. It is extensively explained in section 1.3.2

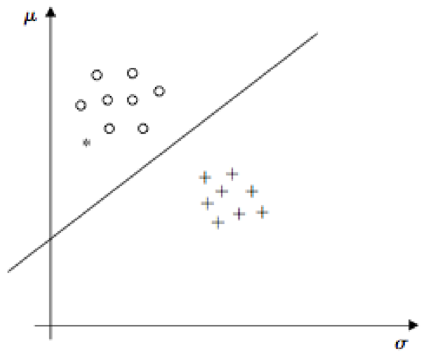

Figure 1.2: Plot of the mean value versus the standard deviation for a number of different images originating from class A (circles) and class B (crosses). In this case, a straight line separates the two classes

The preceding artificial classification task has outlined the rationale behind a large class of pattern recognition problems. The measurements used for the classification, the mean value and the standard deviation in this case, are known as features. In the more general case M features x_m, $m = 1, 2, \ldots, M$, are used, and they form the feature vector

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}$$

Each of the feature vectors identifies uniquely a single pattern (object). Throughout this work features and feature vectors will be treated as random variables and vectors, respectively. This is natural, as the measurements resulting from different patterns exhibit a random variation. This is due partly to the measurement noise of the measuring devices and partly to the distinct characteristics of each pattern. For example, in X-ray imaging large variations are expected because of the differences in physiology among individuals. This is the reason for the scattering of the points in each class shown in Fig. 1.1. Notice that a large variety of features can be selected depending on the application they are used for. In our case, as we will see later, since our goal is spatial clustering, we work with images. Nevertheless, the objects are not the images themselves, but are the pixels. More about that in section 1.4.

The straight line in Fig. 1.2 is known as the decision line, and it constitutes the classifier whose role is to divide the feature space into regions that correspond to either class A or class B. If a feature vector $\boldsymbol{x}$, corresponding to an unknown pattern, falls in the class A region, it is classified as class A, otherwise as class B. This does not necessarily mean that the decision is correct. If it is not correct, a misclassification has occurred. In order to draw the straight line in Fig. 1.2 we exploited the fact that we knew the labels (class A or B) for each point of the figure.

The patterns (feature vectors) whose true class is known and which are used for the design of the classifier are known as training patterns (training feature vectors). After that, the classifier will be evaluated with the test patterns (test feature vectors), of which we know the labels, but they are not used for designing the classifier.

In the unsupervised case, widely explained in Chapter 2, the true labels are unknown, therefore the whole data set is used in the classification procedure, called clustering procedure in those circumstances.

1.3 The Steps in a Classification Task

1.3.1 Stages

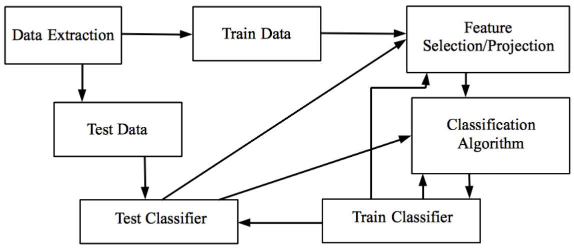

Figure 1.3: Steps to design the classifier

Having outlined the definitions and the rationale, let us point out the basic questions arising in a classification task.

• How are the features generated?

Data adcquisition: there are several ways to acquire the data vectors.

- Sensors: it consists in the acquisition of the pattern to analyze using some transducer (cameras, microphones, etc.), to have a representation in a digital format.
- Data base.
- Generate our own data: sometimes, if we want to test an algorithm, we can create the data in a way we think it will be useful.

After data acquisition some different pre-processing task can be performed, like noise removal, normalization, etc.

Train and test subsets: split up the full set $\boldsymbol{Y}$ of size $M \times N$, M-dimensional feature vectors describing N objects. If we use all data for training and the same data for testing, we might overtrain the classifier. There are different methods, for example resubstitution, hold-out, cross-validation or bootstrap[33].

• What is the best number M of features to use?

Feature selection/extraction: it consists in a stage where the data are reduced representing them with different qualitative or quantitative features or characteristics. Feature selection refers to selecting a subset of the available ones without transformation, whereas in feature extraction new features are created using combinations or transformations of the existing feature set.

Feature selection/extraction is a complex matter, and it is not the aim of this Thesis going into detail on it. Anyway, emphasize that there are different, and well known, methods as PCA, LDA, for feature projection, which create a subset of new features by combinations of the existing ones. However, most of the times the optimal feature map is a non-linear one, in that case the Kernel-PCA and Kernel-LDA are the suitable ones.

• Having adopted the appropriate, for the specific task, features, how does one design the classifier?

Classification: at this stage the features are analyzed to determine to what class or category the object belongs. The features should discriminate among the different classes to recognize and be similar to the objects at the same class.

In the preceding example the straight line was drawn empirically, just to please the eye. In practice, this cannot be the case, and the line should be drawn optimally, with respect to an optimality criterion. Furthermore, problems for which a linear classifier (straight line or hyperplane in the M-dimensional space) can result in acceptable performance are not the rule. In general, the surfaces dividing the space in the various class regions are nonlinear. What type of nonlinearity must one adopt, and what type of optimizing criterion must be used in order to locate a surface in the right place in the M-dimensional feature space? These questions concern the classifier design stage.

• Finally, once the classifier has been designed, how can one assess the performance of the designed classifier?

Evaluation: this part evaluates the cost of the decision of the classifier and tries to reduce the risk of errors using different tools, basically from the context of our observations.

In real applications, these stages may not be so differentiated, merging different functions in some stages. Furthermore, the working mode may not be only unidirectional as posterior stages can report different results to adapt the behavior of earlier sections, see Fig. 1.3.

1.3.2 Data Representation: Scatter Plot

One of the most useful ways to represent the data is the *scatter plot*. In fact, we use it to represent our data in Chapter 5, hence we think it may be useful to explain what is a scatter plot.

A scatter plot or scattergraph is a type of mathematical diagram using cartesian coordinates to display values for two variables for a set of data. The data is displayed as a collection of points, each having the value of one variable determining the position on the horizontal axis and the value of the other variable determining the position on the vertical axis. This kind of plot is also called a scatter chart, scatter diagram and scatter graph. Scatter plot is used when a variable exists that is under the control of the experimenter. If a parameter exists that is systematically incremented and/or decremented by the other, it is called the control parameter or independent variable and is customarily plotted along the horizontal axis. The measured or dependent variable is customarily plotted along the vertical axis. If no dependent variable exists, either type of variable can be plotted on either axis and a scatter plot will illustrate only the degree of correlation (not causation) between two variables.

A scatter plot can suggest various kinds of correlations between variables with a certain confidence interval. Correlations may be positive (rising), negative (falling), or null (uncorrelated). If the pattern of dots slopes from lower left to upper right, it suggests a positive correlation between the variables being studied. If the pattern of dots slopes from upper left to lower right, it suggests a negative correlation. A line of best fit (alternatively called 'trendline') can be drawn in order to study the correlation between the variables. An equation for the correlation between the variables can be determined by established best-fit procedures. For a linear correlation, the best-fit procedure is known as linear regression and is guaranteed to generate a correct solution in a finite time. Unfortunately, no universal best-fit procedure is guaranteed to generate a correct solution for arbitrary relationships.

A scatter plot is also very useful when we wish to see how two comparable data sets agree with each other. One of the most powerful aspects of a scatter plot, however, is its ability to show nonlinear relationships between variables. In addition, if the data is represented by a mixture model of simple relationships, these relationships will be visually evident as superimposed patterns.

1.3.3 Evaluation

During the experimental work, the developed method is evaluated by some different evaluation measures. Therefore, we think that is necessary to introduce some of these measures. For this purpose, some supervised measures are presented below, and in Chapter 2, some methods for evaluating the unsupervised clustering validity are explained.

Confusion Matrix

To find out how the errors are distributed across the classes we construct a confusion matrix[33] using the testing data set, $\boldsymbol{Y}$. The entry a_{ij} of such a matrix denotes the number of elements from $\boldsymbol{Y}$ whose true class is ω_i, and which are assigned by the algorithm to class ω_j.The estimate of the algorithms accuracy can be calculated as the trace of the matrix divided by the total sum of the entries (N). The additional information that the confusion matrix provides is where the misclassifications have occurred.

In a two class example the confusion matrix is defined as:

		TRUE	
		Positive	**Negative**
ESTIMATED	**Positive**	TP	FP
	Negative	FN	TN

Figure 1.4: Confusion Matrix, 2 class example

From this information several rates can be calculated:

· Accuracy:
$P(correct) = (TP + TN)/N$.

· Sensitivity:
$P(correct\ positive) = TP/(TP + FN)$.

· Specificity:
$P(correct\ negative) = TN/(TN + FP)$.

· Positive precision:
$P(positive|positive) = TP/(TP + FP)$

· Negative precision:
$P(negative|negative) = TN/(TN + FN)$

· False alarm rate:
$P(FA) = FP/(FP + TN)$

Dice Coefficient

Dice coefficient[61], named after Lee Raymond Dice, is a similarity measure related to the Jaccard index[2].

For sets $\boldsymbol{X}$ and $\boldsymbol{Y}$, the coefficient may be defined as twice the shared information (intersection, in image segmentation the pixels that are equal classified in both segmentations), over the combined set (union):

$$DC = \frac{2|\boldsymbol{X} \cap \boldsymbol{Y}|}{|\boldsymbol{X}| + |\boldsymbol{Y}|} \tag{1.1}$$

where $\boldsymbol{X}$ is the segmentation result, $\boldsymbol{Y}$ is the ground truth, and $|*|$ denotes the number of pixels contained in a set.

It can be considered as a overlap metric for clustering evaluation. A value of 0 indicates no overlap; a value of 1 indicates perfect agreement.

[2]The Jaccard index[5], or similarity coefficient, is a statistic used for comparing the similarity and diversity of sample sets.

Higher numbers indicate better agreement, and in the case of segmentation indicate that the results match the gold standard better than results that produce lower Dice coefficients.

In terms of false positive, false negative, true negative, and true positive counts, these numbers may also be written as $2TP/((FP+TP)+(TP+FN))$.

Of course, there are many more evaluation measures, for example, another well known one is the ROC; as well as the unsupervised validity measures (explained in Chapter 2). But, in this Chapter we only have explained the ones that we use in our experimental work, since we do not want to overextend.

1.4 Supervised, Unsupervised and Semi-supervised Learning

In the example of Fig. 1.1, we assumed that a set of training data were available, and the classifier was designed by exploiting this a priori known information. This is known as supervised pattern recognition or in the more general context of machine learning as supervised learning. However, this is not always the case, and there is another type of pattern recognition tasks for which training data, of known class labels, are not available. In this type of problem, we are given a set of feature vectors $\boldsymbol{x}$ and the goal is to unravel the underlying similarities and cluster (group) 'similar' vectors together. This is known as unsupervised pattern recognition or unsupervised learning or clustering. Such tasks arise in many applications in social sciences and engineering, such as remote sensing, image segmentation (our case), and image and speech coding. Let us pick two such problems, which are able to better explain the application we work with: multispectral image segmentation.

In multispectral remote sensing, the electromagnetic energy emanating from the earths surface is measured by sensitive scanners located aboard a satellite, an aircraft, or a space station. This energy may be reflected solar energy (passive) or the reflected part of the energy transmitted from the vehicle (active) in order to 'interrogate' the earths surface. The scanners are sensitive to a number of wavelength bands of the electromagnetic radiation. Different properties of the earths surface contribute to the reflection of the energy in the different bands. For example, in the visibleinfrared range properties such as the mineral and moisture contents of soils, the sedimentation of water, and the moisture content of vegetation are the main contributors to the reflected energy. In contrast, at the thermal end of the infrared, it is the thermal capacity and thermal properties of the surface and near subsurface that contribute to the reflection. Thus, each band measures different properties of the same patch of the earths surface. In this way, images of the earths surface corresponding to the spatial distribution of the reflected energy in each band can be created. The task now is to exploit this information in order to identify the various ground cover types, that is, built-up land, agricultural land, forest, fire burn, water, and diseased crop. To this end, one feature vector $\boldsymbol{x}$ for each cell from the 'sensed' earths surface is formed.

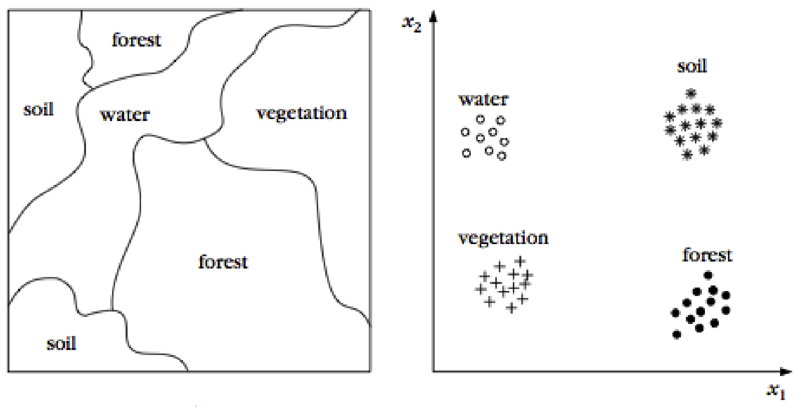

Figure 1.5: An illustration of various types of ground cover (left) and clustering of the respective features for multispectral imaging using two bands (right)

The elements $x_m, m = 1, 2, \ldots, M$, of the vector are the corresponding image pixel intensities in the various spectral bands. In practice, the number of spectral bands varies.

A clustering algorithm can be employed to reveal the groups in which feature vectors are clustered in the M-dimensional feature space. Points that correspond to the same ground cover type, such as water, are expected to cluster together and form groups. Once this is done, the analyst can identify the type of each cluster by associating a sample of points in each group with available reference ground data, that is, maps or visits. Fig.1.5 demonstrates the procedure. Clustering is also widely used in the social sciences in order to study and correlate survey and statistical data and draw useful conclusions, which will then lead to the right actions. Let us again resort to a simplified example and assume that we are interested in studying whether there is any relation between a countrys gross national product (GNP) and the level of peoples illiteracy, on the one hand, and childrens mortality rate on the other. In this case, each region is represented by a three-dimensional feature vector whose coordinates are indices measuring the quantities of interest. A clustering algorithm will then reveal a rather compact cluster corresponding to regions that exhibit low GNPs, high illiteracy levels, and high childrens mortality expressed as a population percentage.

In this Thesis, we work with a similar type of data. A set of M multispectral images (computer simulated or real) are provided. Our task consists of classifying each pixel into one class or another. In this case, if we have an $I \times J$ image, we have $I \times J$ pixels, therefore $I \times J$ samples/observations, each of which has M features, corresponding to the intensity value at this pixel location in the M different images. These intensity values can mean any type of feature that can be extracted from the image, from color (if it is a picture) to density population (if it is a map).

A major issue in unsupervised pattern recognition is that of defining the 'similarity' between two feature vectors and choosing an appropriate measure for it. Another issue of importance is choosing an algorithmic scheme that will cluster (group) the vectors on the basis of the adopted similarity measure. In general, different algorithmic schemes may lead to different results, which the expert has to interpret.

Finally, semi-supervised learning/pattern recognition for designing a classification system shares the same goals as the supervised case, however now, the designer has at his or her disposal a set of patterns of unknown class origin, in addition to the training patterns, whose true class is known. We usually refer to the former ones as unlabeled and the latter as labeled data. Semi-supervised pattern recognition can be of importance when the system designer has access to a rather limited number of labeled data. In such cases, recovering additional information from the unlabeled samples, related to the general structure of the data at hand, can be useful in improving the system design. Semi-supervised learning finds its way also to clustering tasks. In this case, labeled data are used as constraints in the form of must-links and cannot-links. In other words, the clustering task is constrained to assign certain points in the same cluster or to exclude certain points of being assigned in the same cluster. From this perspective, semi-supervised learning provides an a priori knowledge that the clustering algorithm has to respect.

Since the aim of this work is Image Segmentation, in the following section we introduce some basic knowledge about it as a specific case of a pattern recognition problem.

1.5 Image segmentation

1.5.1 Introduction

Image segmentation is of great interest in a variety of scientific and industrial areas, with applications in biomedicine, remote sensing, control of quality and many others. The main purpose of image segmentation is to extract information from the images to distinguish different objects of interest. Several methods for supervised and unsupervised image segmentation and classification have been proposed in the past.

A segmentation procedure usually consists of two steps. The first step is to choose a proper set of features which can identify the same-content regions and meanwhile differentiate different-content regions; the second step is to apply a segmentation method to the chose features to achieve a segmentation map. Imaginary in different applications varies and may require different methods to extract distinct features. Feature extraction is a broad topic and this work is focused only on how to develop segmentation methods assuming the features used are sufficient to identify same-content regions and differentiate different-content regions.

1.5.2 Types and Background

Image segmentation plays an important role in image analysis and computer vision, which is also regarded as the bottleneck of the development of image processing technology. The goal of image segmentation is partition of an image into a set of disjoint regions with uniform and homogeneous attributes. The image segmentation approaches can be divided into four categories[55]: thresholding, clustering, edge detection, and region extraction. In this work, a clustering based method will be considered.

Clustering is a process for classifying objects or patterns in such a way that samples of the same group are more similar to one another than samples belonging to different groups.

Many clustering strategies have been used, such as the hard clustering scheme and the fuzzy clustering scheme, each of which has its own special characteristics. The conventional hard clustering method restricts each point of the data set to exclusively just one cluster. As a consequence, with this approach the segmentation results are often very crisp, i.e., each pixel of the image belongs to exactly just one class. However, in many real situations, for images, issues such as limited spatial resolution, poor contrast, overlapping intensities, noise and intensity inhomogeneities variation make this hard (crisp) segmentation a difficult task.

Due to the fuzzy, set theory[57] was proposed, which produced the idea of partial membership of belonging described by a membership function, fuzzy clustering as a soft segmentation method has been widely studied and successfully applied in image segmentation. Among the fuzzy clustering methods, fuzzy C-means (FCM) algorithm[9] is the most popular method used in image segmentation because it has robust characteristics for ambiguity and can retain much more information than hard segmentation methods. Although the conventional FCM algorithm works well on most noise-free images, it has a serious limitation, i.e., it does not incorporate any in formation about spatial context, which cause it to be sensitive to noise and imaging artifacts.

To compensate this drawback of FCM, the obvious way is to smooth the image before segmentation. However, the conventional smoothing filters can result in loss of important image details, especially boundaries or edges of image. More importantly, there is no way to rigorously control the trade-off between the smoothing and clustering.

Recently, fuzzy logic and statistics describing the uncertainty of segmentation and classification are used for segmentation tasks.

Salzenstein and Pieezynski[49] firstly introduced Fuzzy Markov random field (FMRF) to the field image segmentation, Ruan and Bloyet[48] combined fuzzy MRFs and stochastic approaches for segmentation of magnetic resonance images; Lu and Zhou[39]. applied multi-level FMRF to synthetic aperture radar images segmentation.

In this work a improvement to fuzzy clustering is described. A prior spatial constraint is introduced into FCM algorithm through Markov random field theory and its equivalent Gibbs random field theory, in which the spatial information is encoded through mutual influences of neighboring sites. In Chapter 4, we present the method which estimates the parameters defining the Markovian distribution of the measured data while performing the data clustering simultaneously.

1.6 Report Structure

After this brief introduction to pattern recognition, this work has been divided into six more chapters:

Chapter 2 introduces the most relevant notions about cluster analysis, as well as, the main clustering algorithms. Then, some of the unsupervised cluster validation methods are explained.

Chapter 3 presents, first, the spatial clustering methods. Next, it describes the spatial fuzzy clustering and the first methods based on this approach, in contrast to the hard clustering methods. Then, the theory of Markov random fields and its applications in spatial clustering are explained. Finally, the relationship between these methods is exposed.

Chapter 4 describes and develops the proposed method, a spatial fuzzy clustering algorithm with simultaneous estimation of Markov random field parameters and class.

Chapter 5 contains the experimental work of the thesis. It presents the data used in the experiments, along with the results obtained.

This work is only a little part of a more complex project, which is being developed in support of crime prediction/prevention. Right now, the Illinois Institute of Technology is working with the Chicago Police Department (CPD) in order to develop an automated, proactive predictive analytics system. Hence, in Chapter 6 a brief survey of this project is given. It also describes the software used for this purpose and the first experiments that have been conducted.

Finally, Chapter 7 makes a summary and discusses the results obtained in the experiments. The most important conclusions are extracted, and future work directions are suggested.

Being all the simulations, algorithms and experiments written in Matlab code, we think that including in the appendix the code of the proposed method may be interesting.

Chapter 2

Unsupervised Learning

In this Chapter we introduce the Unsupervised learning, which include methods that do not rely on predefined class and class-labeled training examples

The pattern recognition methods focused on the issue of classification, where a pattern consisted of a pair of variables $\boldsymbol{x}$, a collection of observations or features (feature vector), and ω, the concept behind the observation (label), are called supervised (training with a teacher), since the system is given both the feature vector and the correct answer. In contrast, unsupervised methods operate on unlabeled data, given a collection of feature vectors $\boldsymbol{Y} = (\boldsymbol{x}_1\ \boldsymbol{x}_2\ \ldots\ \boldsymbol{x}_N)$ without class labels ω_k. Inasmuch as they are not provided with the correct answer, the class labels are not known and the number of classes may not been known either, these methods attempt to build a model that captures the structure of the data.

The supervised and unsupervised paradigms comprise the vast majority of pattern recognition problems[1]. Although unsupervised learning methods may appear to have limited capabilities, there are several reasons that make them extremely useful. One of the main reasons for using unsupervised methods is the fact that either labeling large data sets can be a costly procedure (i.e., speech recognition),or class labels may not be known beforehand (i.e., data mining).

A first classification of the unsupervised learning methods is:

Parametric (mixture models)**:** these methods model the underlying class-conditional densities with a mixture of parametric densities, and the objective is to find the model parameters

$$P(\boldsymbol{x}|\boldsymbol{\theta}) = \sum P(\boldsymbol{x}|\omega, \boldsymbol{\theta})P(\omega)$$

The parametric methods are closely related to parameter estimation. If we look at Fig. 2.1 and Fig. 2.2, it can be observed that the steps followed by the parametric unsupervised methods are quite similar to the ones followed by the supervised learning methods. One of the main differences is the fact that in supervised methods the dataset is split in train and test

[1]A third approach, known as reinforcement learning, uses a reward signal (real- valued or binary) to tell the learning system how well it is performing. In reinforcement learning, the goal of the learning system (or agent) is to learn a mapping from states onto actions (an action policy) that maximizes the total reward

datasets; that would be meaningless in an unsupervised method, since the true categories are not known in order to test the algorithms. It is important to notice that, in the parametric unsupervised methods, through the use of EM algorithm, explained in subsection 2.1.7, the identity of the component that originated each data point was treated as a missing feature. The solution to the mixture problem (the EM algorithm) is also used for Hidden Markov Models, which is quite related to the method proposed in this work, as we will see in Chapter 3.

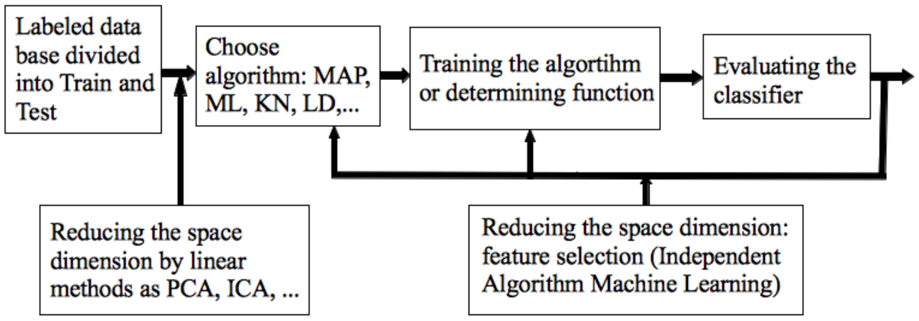
Labeled data base divided into Train and Test
Choose algorithm: MAP, ML, KN, LD,...
Training the algortihm or determining function
Evaluating the classifier
Reducing the space dimension by linear methods as PCA, ICA, ...
Reducing the space dimension: feature selection (Independent Algorithm Machine Learning)

Figure 2.1: Steps for Supervised Learning methods

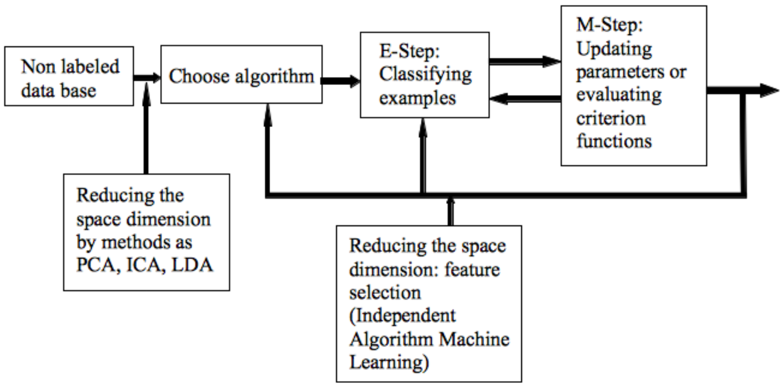
Non labeled data base
Choose algorithm
E-Step: Classifying examples
M-Step: Updating parameters or evaluating criterion functions
Reducing the space dimension by methods as PCA, ICA, LDA
Reducing the space dimension: feature selection (Independent Algorithm Machine Learning)

Figure 2.2: Steps for Parametric Unsupervised Learning methods

A particular form of the mixture model problem leads to the most widely used clustering method: the k-means algorithm.

Non-parametric (clustering)**:** No assumptions are made about the underlying densities. Instead, we are concerned with finding natural groups (clusters) in a dataset. Non-parametric clustering involves three steps: (1) defining a measure of (dis)similarity between examples, (2) defining a criterion function for clustering, (3) defining an algorithm to optimize the criterion function. These methods are typically referred to as clustering, and they can be focused on statistical clustering or on connectionist approaches

2.1 Cluster Analysis

Clustering can be considered the most important unsupervised learning problem; so, as every other problem of this kind, it deals with finding a structure in a collection of unlabeled data. A loose definition of clustering could be *the process of organizing objects into groups whose members are similar in some way.* A cluster is therefore a collection of objects which are *similar* between them and are *dissimilar* to the objects belonging to other clusters. Therefore, the aim of cluster Analysis is to find the similarities between data according to the characteristics found in the data and grouping similar data objects into clusters.

There are different types of clustering depending on the similarity criterion. For example, the criterion could be distance, two or more objects belong to the same cluster if they are close according to a given distance (e.g. geometrical distance); this is called distance-based clustering. Another kind of clustering is conceptual clustering, two or more objects belong to the same cluster if this one defines a concept common to all that objects, in other words, objects are grouped according to their fit to descriptive concepts, not according to simple similarity measures.

It can be shown that there is no absolute best criterion which would be independent of the final aim of the clustering. Consequently, it is the user which must supply this criterion, in such a way that the result of the clustering will suit their needs. For instance, we could be interested in finding representatives for homogeneous groups (data reduction), in finding 'natural clusters' and describe their unknown properties (natural data types), in finding useful and suitable groups (useful data classes) or in finding unusual data objects (outlier detection).

The typical applications of Clustering algorithms are as a stand-alone tool to get insight into data distribution or as a preprocessing step for other algorithms. It can be applied in many fields: Marketing, finding groups of customers with similar behavior given a large database of customer data containing their properties and past buying records; Biology, classification of plants and animals given their features; Libraries, book ordering; Insurance, identifying groups of motor insurance policy holders with a high average claim cost, identifying frauds; City-planning, identifying groups of houses according to their house type, value and geographical location; Earthquake studies, clustering observed earthquake epicenters to identify dangerous zones; www, document classification, clustering weblog data to discover groups of similar access patterns.

The main requirements that a clustering algorithm should satisfy are: scalability, dealing with different types of attributes, discovering clusters with arbitrary shape, minimal requirements for domain knowledge to determine input parameters, ability to deal with noise and outliers, insensitivity to order of input records, high dimensionality, interpretability and usability.

A good clustering method will produce high quality clusters in which: the intra-class (that is, intra-cluster) similarity is high, the inter-class similarity is low. The quality[24] of a clustering result also depends on both the similarity measure used by the method and its implementation, its ability to discover some or all of the hidden patterns, and the definition and representation of cluster chosen.

There are a number of problems with clustering. Among them:

- Current clustering techniques do not address all the requirements suitably (and concurrently).
- Dealing with large number of dimensions and large number of data items can be problematic because of time complexity.
- The effectiveness of the method depends on the definition of *distance* (for distance-based clustering).
- If an obvious distance measure does not exist we must define it, which is not always easy, especially in multi-dimensional spaces.
- The result of the clustering algorithm (that in many cases can be arbitrary itself) can be interpreted in different ways.
- We must select K.

2.1.1 Distance Measure

Types of data in Cluster Analysis[24]: Interval-scaled variables , binary variables, nominal-ordinal-ratio variables, variable of mixed types. All of them can be treated as Interval-scaled variables after a suitable transformation.

We have the data matrix and the dissimilarity matrix[25]. The similarity/dissimilarity is usually expressed in terms of a distance function, typically metric: $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The definitions of distance functions are usually very different for interval-scaled, boolean, categorial, ordinal ratio, and vector variables. It is hard to define 'similar enough' or 'good enough, the answer is typically highly subjective. Thus, an important step in most clustering is to select a distance measure, which will determine how the similarity of two elements is calculated. This 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. Typical alternatives to calculate the distance between clusters are:

Single link: the smallest distance between element in one cluster and element in the other, e.g., Single Linkage Agglomerative Hierarchical clustering, it requires only a single dissimilarity to be small but the produced clusters can violate the'compactness' property (cluster with large diameters).

Complete link: the largest distance between an element in one cluster and an element in the other, e.g., Complete Linkage Agglomerative Hierarchical clustering, opposite extreme, compact clusters with small diameters, but can violate the closeness' property (observations assigned to a cluster can be more closer to members of other clusters).

Average: the averaged distance between an element in one cluster and an element in the other, e.g., Group Average Hierarchical clustering, compromise, it attempts to produce relatively compact clusters and relatively far apart, but it depends on the dissimilarity scale.

Centroid: the distance between the centroids of two clusters, e.g., K-means clustering.

Medoid: distance between the medoids of two clusters, where the medoid is one object chosen that is centrally located in the cluster, e.g., K-medoids.

Another important distinction is whether the clustering uses symmetric or asymmetric distances. Many of the distance functions listed above have the property that distances are symmetric (the distance from object A to B is the same as the distance from B to A). Notice that a true metric gives symmetric measures of distance.

Definition of a metric: a measuring rule $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for the distance between two vectors $\boldsymbol{x}_i, \boldsymbol{x}_j$ is considered a metric if it satisfies the following properties:

$$\begin{aligned}
&d(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq d_0 \\
&d(\boldsymbol{x}_i, \boldsymbol{x}_j) = d_0 \qquad \iff \boldsymbol{x}_i = \boldsymbol{x}_j \\
&d(\boldsymbol{x}_i, \boldsymbol{x}_j) = d(\boldsymbol{x}_j, \boldsymbol{x}_i) \\
&d(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq d(\boldsymbol{x}_i, \boldsymbol{x}_k) + d(\boldsymbol{x}_k, \boldsymbol{x}_j)
\end{aligned}$$

if the metric has also the property $d(a\boldsymbol{x}_i, a\boldsymbol{x}_j) = |a| \cdot d(\boldsymbol{x}_i, \boldsymbol{x}_j)$, then it is called a norm and denoted $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = ||\boldsymbol{x}_i - \boldsymbol{x}_j||$.
The most general form of a distance metric is the **power norm**[22]:

$$||\boldsymbol{x}_i - \boldsymbol{x}_j||_{p/r} = \left(\sum_{m=1}^{M} |x_{mi} - x_{mj}|^p \right)^{1/r} \tag{2.1}$$

where parameter p controls the weight placed on any dimension dissimilarity, whereas parameter r controls the distance growth of patterns that are further apart. Notice that the definition of norm must be relaxed, allowing a power factor for $|a|$.

Most of the commonly used metrics are derived from the power norm:

Minkowski (L_k norm)**:**

$$||\boldsymbol{x}_i - \boldsymbol{x}_j||_k = \left(\sum_{m=1}^{M} |x_{mi} - x_{mj}|^k \right)^{1/k} \tag{2.2}$$

The choice of an appropriate value of k depends on the amount of emphasis that you would like to give to the larger difference between dimensions.

Manhattan or **city-block distance** (L_1 norm)**:**

$$||\boldsymbol{x}_i - \boldsymbol{x}_j||_{c-b} = \sum_{m=1}^{M} |x_{mi} - x_{mj}| \tag{2.3}$$

When used with binary vectors, the L_1 norm is known as the Hamming distance.

Euclidean norm (L_2 norm)**:**

$$||\boldsymbol{x}_i - \boldsymbol{x}_j||_e = \left(\sum_{m=1}^{M} |x_{mi} - x_{mj}|^2 \right)^{1/2} \tag{2.4}$$

Chebyshev distance (L_∞ norm)**:**

$$||\boldsymbol{x}_i - \boldsymbol{x}_j||_c = \max_{1 \leq m \leq M} |x_{mi} - y_{mj}| \tag{2.5}$$

Other metrics are also popular:

Quadratic distance:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{\Sigma} (\boldsymbol{x}_i - \boldsymbol{x}_j)} \tag{2.6}$$

The Mahalanobis distance is a particular case of this distance:

$$d_M^2(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{\Sigma} (\boldsymbol{x}_i - \boldsymbol{x}_j) \tag{2.7}$$

Canberra metric for non-negative features**:**

$$d_{ca}(vectx_i, \boldsymbol{x}_j) = \sum_{m=1}^{M} \frac{|x_{mi} - x_{mj}|}{x_{mi} + x_{mj}} \tag{2.8}$$

Non-linear distance:

$$d_N(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{cases} 0 & if \quad d_e(\boldsymbol{x}_i, \boldsymbol{x}_j) < T \\ H & if \quad d_e(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq T \end{cases} \tag{2.9}$$

where T is a threshold and H is a distance.

Notice that the above distance metrics are measures of dissimilarity, but some measures of similarity also exist:

Inner product:

$$s_{\text{inner}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T \boldsymbol{x}_j \tag{2.10}$$

The inner product is used when the vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are normalized, so that they have the same length.

Correlation coefficient:

$$s_{CC}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\sum_{m=1}^{M} (x_{mi} - \mu_{mi})(x_{mj} - \mu_{mj})}{\left[\sum_{m=1}^{M} (x_{mi} - \mu_{mi})^2 \sum_{m=1}^{M} (x_{mj} - \mu_{mj})^2 \right]^{\frac{1}{2}}} \tag{2.11}$$

Tanimoto measure (for binary-valued vectors)**:**

$$s_T(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\boldsymbol{x}_i^T \boldsymbol{x}_j}{||\boldsymbol{x}_i||^2 + ||\boldsymbol{x}_j||^2 - \boldsymbol{x}_i^T \boldsymbol{x}_j} \tag{2.12}$$

Once a (dis)similarity measure has been determined, we need to define a criterion function to be optimized. the most widely used criterion function for clustering is the sum-of-square error:

$$J_{MSE} = \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in \omega_k} ||\boldsymbol{x} - \boldsymbol{\mu}_k||^2 \tag{2.13}$$

where $\quad \boldsymbol{\mu}_k = \dfrac{1}{N} \sum_{\boldsymbol{x} \in \omega_k} \boldsymbol{x}$

This criterion measures how well the data set $\boldsymbol{Y} = (\boldsymbol{x}_1\ \boldsymbol{x}_2\ \ldots\ \boldsymbol{x}_N)$ is represented by the cluster centers $\boldsymbol{\mu} = (\boldsymbol{\mu}_1\ \boldsymbol{\mu}_2\ \ldots\ \boldsymbol{\mu}_K)(K < N)$. Clustering methods that use this criterion are called minimum variance methods. Other criterion functions exist, based on scatter matrices used in Linear Discriminant Analysis. the choice of (dis)similarity measure and criterion function will have a major impact on the final clustering produced by the algorithms. notice that the validity of the final cluster solution is highly subjective, in contrast with supervised training, where a clear objective function is known[2]. A number of quantitative methods for cluster validity are proposed in '*Theodoridis and Koutroumbas, 1999*'[52].

Iterative Optimization: once a criterion function has been defined, we must find a partition of the data set that minimizes the criterion. Exhaustive enumeration of all partitions, which guarantees he optimal solution is unfeasible[3]. The common approach is to proceed in an iterative fashion finding some reasonable initial partition and then, moving samples from one cluster to another in order to reduce the criterion function. These iterative methods produce suboptimal solutions but are computationally tractable.

2.1.2 Combinational Algorithms

Without a regard to probability model, directly specify a mathematical loss function and attempt to minimize it though some combinational optimization algorithm:

Assuming a symmetric distance $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = d(\boldsymbol{x}_j, \boldsymbol{x}_i)$,

$$T = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{2}\sum_{k=1}^{K}\sum_{\boldsymbol{x}_i \in \omega_k}\left(\sum_{\boldsymbol{x}_j \in \omega_k} d(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{\boldsymbol{x}_j \notin \omega_k} d(\boldsymbol{x}_i, \boldsymbol{x}_j)\right) \tag{2.14}$$

where given a data set T is the total point scatter, a constant independent of the cluster assignment.

$$T = W(K) + B(K) \tag{2.15}$$

$$W(K) = \frac{1}{2}\sum_{k=1}^{K}\sum_{\boldsymbol{x}_i \in \omega_k}\sum_{\boldsymbol{x}_j \in \omega_k} d(\boldsymbol{x}_i, \boldsymbol{x}_j) \qquad \text{Within Cluster distance} \tag{2.16}$$

$$B(K) = \frac{1}{2}\sum_{k=1}^{K}\sum_{\boldsymbol{x}_i \in \omega_k}\sum_{\boldsymbol{x}_j \notin \omega_k} d(\boldsymbol{x}_i, \boldsymbol{x}_j) \qquad \text{Between Cluster distance} \tag{2.17}$$

Combinational cluster analysis wants to minimize $W(C)$ or maximize $B(C)$ over all the possible assignments of the N points to K clusters. Unfortunately, such optimization is feasible only for very small data sets. For this reason, practical clustering algorithms are able to examine only a very small fraction of all possible encodes. Such feasible strategies are based on iterative gradient descent.

An initial partition in specified, and at each iterative step the cluster assignments are changed in such way that the value of the criterion is improved. When

[2]Bayes Risk.

[3]For example, a problem with 5 clusters and 100 examples yields 10^{67} no.

the prescription is unable to provide an improvement, the algorithm terminates. However these algorithms converge to a local optima.

Clustering techniques have been studied extensively in statistics, machine learning, and data mining, with many methods proposed and studied. These method can be classified into five approaches[58]: partitioning algorithms, hierarchical algorithms, density-based method, grid-based method ,and model-based method. In the following subsections a short explanation of them is given. We test some of these methods in Chapter 5 in order to compare them with the method that we present in this work. That is the reason why we have implemented our own code and there is a deeper explanation of these algorithms.

2.1.3 Partitioning Clustering

Partitioning clustering consists of constructing various partitions an then evaluate them by some criterion.In other words, we want to optimize an objective function by arbitrarily choosing$K$ centers as the initial solution, compute membership of each object according to the present solution, update cluster centers according to the new memberships of the objects until no change of the objective function. Most of this methods rely (implicitly or explicitly) upon estimates of within- and between-cluster scatter matrices.

· Weakness
Although they are widely used, they have some drawbacks as the requirement to specify the parameter K, the inability to find arbitrarily shaped clusters, and the possibility of falling in a local optima.

K-means clustering

K-means[42] (MacQueen, 1967) is an heuristic method and one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume K clusters) fixed *a priori*. The main idea is to define K centroids (typically they are defined choosing randomly k data samples), one for each cluster. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early grouping is done. At this point we need to recalculate K new centroids as barycenters of the clusters resulting from the previous step. After we have these K new centroids, a new binding has to be done between the same data set points and the nearest new centroid. 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. Lastly , this algorithm aims at minimizing an objective function, in this case a squared error function (as we said in Eq. 2.13).

$$J = \sum_{k=1}^{K} \sum_{i=1}^{N} ||\boldsymbol{y}_i^{(k)} - \boldsymbol{c}_k||^2 \tag{2.18}$$

where $||\boldsymbol{y}_i^{(k)} - \boldsymbol{c}_k||^2$ is a chosen distance measure between a data point $\boldsymbol{y}_i^{(k)}$ and the cluster centre $\boldsymbol{c}_k$, is an indicator of the distance of the N data points from

their respective cluster centers (squared Euclidian distance, Eq. 2.4).
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. It often terminates at a local optimum (the global optimum may be found using techniques as deterministic annealing and genetic algorithms). Other weaknesses are the need to specify K number of clusters and the fact that it is not a suitable method to discover clusters with non-convex shapes. The algorithm is also significantly sensitive to the initial randomly selected cluster centers. The K-means algorithm can be run multiple times starting with different starting points to reduce this effect.

Though it is a simple, understandable and relatively efficient method, outliers are another problem that K-means has to deal with. Since the largest distances, results also depend on the metric used to measure distances, have higher influence outliers may distort the distribution of the data. To solve it, there is a variation of the K-means method called *K-medoids*. The algorithm is basically the same except that the centers of each cluster are restricted to be one of the observations, therefore, each cluster is represented by one of the objects in the cluster, which is the mols centrally located in this cluster. PAM, the most common K-medoids algorithm, is more robust than K-means in the presence of noise and outliers because a medoid is less influenced by outliers or other extreme values than a means, but does not scale well for large data sets. The CLARA and CLARANS methods are other K-medoids algorithms that deal with larger data sets than PAM does.

Fuzzy C-Means Clustering

The Algorithm Fuzzy C-means (FCM)[42] is a method of clustering which allows one piece of data to belong to two or more clusters. This method (developed by Dunn in 1973 and improved by Bezdek in 1981) is frequently used in pattern recognition. It is based on minimization of the following objective function (also based in the squared Euclidian distance, Eq. 2.4):

$$J_m = \sum_{i=1}^{N}\sum_{k=1}^{K} u_{ik}^{m} ||\boldsymbol{x}_i - \boldsymbol{c}_k||^2, \quad 1 \leq m < \infty \tag{2.19}$$

where m is any real number greater than 1, u_{ik} is the degree of membership of $\boldsymbol{x}_i$ in the cluster k, $\boldsymbol{x}_i$ is the ith of M-dimensional measured data, $\boldsymbol{c}_k$ is the M-dimension center of the cluster, and $||*||$ is any norm expressing the similarity between any measured data and the center. Fuzzy partitioning is carried out through an iterative optimization of the objective function shown above, with the update of membership u_{ik} and the cluster centers $\boldsymbol{c}_k$ by:

$$u_{ik} = \frac{1}{\sum_{j=1}^{K}\left(\frac{||\boldsymbol{x}_i - \boldsymbol{c}_k||}{||\boldsymbol{x}_i - \boldsymbol{c}_j||}\right)^{\frac{2}{m-1}}} \tag{2.20}$$

$$\boldsymbol{c}_k = \frac{\sum_{i=1}^{N} u_{ik}^{m} \cdot \boldsymbol{x}_i}{\sum_{i=1}^{N} u_{ik}^{m}} \tag{2.21}$$

This iteration will stop when $max_{ik}\{|u_{ik}^{(n+1)} - u_{ik}^{(n)}|\} < \varepsilon$, where ε is a termination criterion between 0 and 1, whereas n 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 $\boldsymbol{U} = [u_{ik}]$ matrix, $\boldsymbol{U}^{(0)}$.
2. At n-step: calculate the center vectors $\boldsymbol{C}^{(n)} = [\boldsymbol{c}_k]$ with $\boldsymbol{U}^{(n)}$.
3. Update $\boldsymbol{U}^{(n)}, \boldsymbol{U}^{(n+1)}$.
4. If $||\boldsymbol{U}^{(n+1)} - \boldsymbol{U}^{(n)}|| < \varepsilon$, then STOP, otherwise return to step 2.

· Remarks
As already told, data are bound to each cluster by means of a Membership Function, which represents the fuzzy behavior of this algorithm. To do that, we simply have to build an appropriate matrix named $\boldsymbol{U}$ whose factors are numbers between 0 and 1, and represent the degree of membership between data and centers of clusters.
For a better understanding, we may consider this simple mono-dimensional example. Given a certain data set, suppose to represent it as distributed on an axis.
Looking at Fig. 2.3, we may identify two clusters in proximity of the two data concentrations. We will refer to them using A and B. In the k–means approach, we associated each datum to a specific centroid; therefore, this membership function looked like in Fig. 2.4.

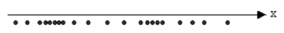

Figure 2.3: mono-dimensional example

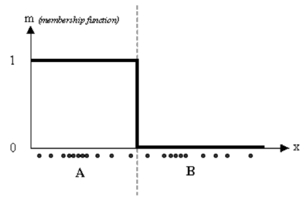

Figure 2.4: k-means membership function

In the FCM approach, instead, the same given datum does not belong exclusively to a well defined cluster, but it can be placed in a middle way. In this case, the membership function follows a smoother line to indicate that every datum may belong to several clusters with different values of the membership coefficient.

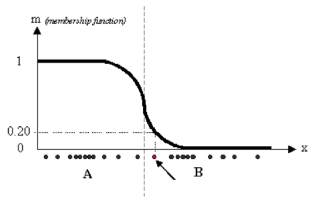

Figure 2.5: FCM membership function

In the Fig. 2.5, the datum shown as a red marked spot belongs more to the B cluster rather than the A cluster. The value 0.2 of m indicates the degree of membership to A for such datum.

Now, instead of using a graphical representation, we introduce a matrix $\boldsymbol{U}$ whose factors are the ones taken from the membership functions:

$$(a) \quad \boldsymbol{U}_{N\times K} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ \multicolumn{2}{c}{\dots\dots} \\ 0 & 1 \end{bmatrix} \qquad\qquad (b) \quad \boldsymbol{U}_{N\times K} = \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \\ 0.6 & 0.4 \\ \multicolumn{2}{c}{\dots\dots\dots} \\ 0.9 & 0.1 \end{bmatrix}$$

The number of rows and columns depends on how many data and clusters we are considering. More exactly we have $K = 2$ columns ($K = 2$ clusters) and N rows, where C is the total number of clusters and N is the total number of data. The generic element is so indicated: $\boldsymbol{u}_{ik}$. In the examples above we have considered the k-means (a) and FCM (b) cases. We can notice that in the first case (a) the coefficients are always unitary. It is so to indicate the fact that each datum can belong only to one cluster. Other properties are shown:

- $u_{ik} \in [0,1] \qquad \forall i,k$
- $\displaystyle\sum_{j=1}^{K} u_{ik} = 1 \qquad \forall i$
- $\displaystyle 0 < \sum_{i=1}^{N} u_{ik} < N \quad \forall N$

2.1.4 Hierarchical Clustering

The Hierarchical clustering method splits a data set into several levels ($N-1$) of partitions (clusters). It requires to specify a measure of dissimilarity (or similarity)[25] based on the pairwise dissimilarity among the observations in two groups, so it uses the distance ($N \times N$)matrix as clustering criteria. The entire hierarchy, which can be represented by a dendogram (a tree which splits the database recursively into smaller subsets), represents an ordered sequence of each group. One advantage that the other methods don't have is that a initial number of clusters is not required but it needs a termination condition; once the algorithm is finished, it is up to the user to decide which level represents a 'natural' clustering. The clustering of the data objects is obtained by cutting the dendogram at the desired level, the Gap statistic can be used for this aim., then each connected component forms a cluster.

The major drawback of the Hierarchical approach is that the entire hierarchy is sensitive to (possibly erroneous) previous cluster merging (or splitting), i.e., data are not permitted to change cluster membership once assignment has taken place.

Hierarchical algorithms can either be created from the leaves up to the root, called Agglomerative, or from the root down to the leaves, called Divisive.

Some examples of Hirerachical clustering methods are the BIRCH, Diana, Agnes, ROCK and CAMALEON[38].

Agglomerative

The Agglomerative Hierarchical clustering is a bottom-up approach. It is based on merging clusters iteratively. Starting with each object forming a separate group, the algorithm merges groups according to some principle: e.g., distance between centers, until a termination condition holds (usually until all objects are in a single cluster).

Several approaches differ only in their definition of between-cluster similarity. Depending on this similarity (or dissimilarity) function they can be classified as Single Linkage, Complete Linkage or Group Average clustering, depending on the alternative to calculate the distance chosen, see subsection **??**. If the data exhibit strong clustering tendency, all 3 methods produce similar results.

Although it is nice that you get a hierarchy instead of an amorphous collection of groups, the major weakness of Agglomerative clustering methods are the time complexity, and, as a Hierarchical method, the fact that what was done previously can never been undone.

Divisive

The Divisive algorithm splits clusters iteratively from the root, thus it is a top-down approach. It starts with all objects in the same cluster, and a cluster is split into smaller clusters (e.g., according to the largest between-group dissimilarity), until a termination condition holds (usually when there is only one object in each cluster).

Although it has a potential advantage over Agglomerative methods when interest is focused on partitioning the data into a relatively small number of clusters, Divisive methods are not generally available, and rarely have been applied.

2.1.5 Density-Based Clustering

The Density-based Clustering is based on density, local cluster criterion, such as density-connected points. The major characteristics of this method are the ability of discovering clusters of arbitrary shape in one scan, the capability of handling noise, and the requirement of density parameters as termination condition. For use this algorithm two parameters have to be defined:

– Eps: maximum radius of the neighborhood.

– MinPts: minimum number of points in an Eps-neighborhood of that point.

$$\boldsymbol{N}_{Eps}(\boldsymbol{p}) : \{\boldsymbol{q} \in \boldsymbol{Z} | dist(\boldsymbol{p}, \boldsymbol{q}) \leq Eps\}$$

A point $\boldsymbol{p}$ is called directly density- reachable from a point $\boldsymbol{q}$, w.r.t Eps, MinPts, if:

– $\boldsymbol{p}$ belongs to $\boldsymbol{N}_{Eps}(\boldsymbol{q})$

– core point condition: $|\boldsymbol{N}_{Eps}(\boldsymbol{q})| \geq Minpts$

While, a point $\boldsymbol{p}$ is density-reachable from a point $\boldsymbol{q}$ wrt. Eps, MinPts if there is a chain of points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n, \boldsymbol{p}_1 = \boldsymbol{q}, \boldsymbol{p}_n = \boldsymbol{p}$ such that $\boldsymbol{p}_i + 1$ is directly density-reachable from $\boldsymbol{p}_i$.
Finally, a point $\boldsymbol{p}$ is density-connected to a point $\boldsymbol{q}$, w.r.t. Eps, MinPts, if there is a point $\boldsymbol{o}$ such that both, $\boldsymbol{p}$ and $\boldsymbol{q}$, are density-reachable from $\boldsymbol{o}$ w.r.t. Eps and MinPts.

Examples: DBSCAN, OPTICS, and Denclue[38].

2.1.6 Grid-Based Clustering

The grid-based clustering uses multi-resolution grid data structure. Using that type of data structure the algorithm enhances the efficiency of clustering, quantizes the space into a finite number of cells, performs clustering on the grid structure by identifying cells that contain more than a number of points and forms clusters connecting these dense cells. One of the strong points of this algorithm is the independentness of the number of data objects, inasmuch as it only depends on the number of cells in each dimension. Although it has a fast processing time, it looses effectiveness as the number of the dimension increase.

Approaches: WaveCluster, CLIQUE, and STING[38].

The objective of segmentation is to partition an image into homogeneous regions such that the segmentation must be complete (i.e every pixel must be in a region), the pixels in a region must be connected, the regions must be disjoint.

2.1.7 Model-Based Clustering

There is another way to deal with clustering problems: a model-based approach, which consists in using certain models for clusters and attempting to optimize the fit between the data and the model. Depending on the approach used in order to optimize this fit, several methods can be used. The most typical methods[24] are:

Conceptual clustering: it is a form of clustering in machine learning which produces a classification scheme for a set of unlabeled objects finding characteristic description for each concept (class). The most popular is COBWEB, a simple method of incremental conceptual learning. It creates a hierarchical clustering in the form of a classification tree, where each node refers to a concept and contains a probabilistic description of that concept. Unfortunately, it has some limitations, as the assumption that the attributes are independent of each other is often to strong because correlation may exists, and that it is not suitable for clustering large data bases because of skewed tree and expensive probability distributions. Another option is CLASSIT.

Neuronal networks: neuronal network approaches represent each cluster as an exemplar, acting as a 'prototype' of the cluster, then new objects are distributed to the cluster whose exemplar is the most similar according to some distance measure. One of the most common method, proposed by Kohonen in 1981, is SOM (soft-organizing feature map), which can be viewed as a nonlinear projection from an M-dimensional input space onto a lower-order, usually 2 to 3 dimensions target space, so that, the distance and proximity relationships are preserved as much as possible. Clustering is performed by having several units competing for the current object, the unit whose weight vector is closest to the current object wins, 'winner-takes-all' fashion, the winner and its neighbors learn by having their weights adjusted. SOM's are believed to resemble processing that can occur in the brain and it is very useful for visualizing high-dimensional data in 2- or 3-D space.

Statistical approach EM**:** or gaussian mixture model (Banfield and Raftery, 1993), it is a probabilistic variant of k-means method. It starts by choosing k seeds, and regarding the seeds as means of gaussian distributions, then iterates over two steps called the estimation step and the maximization step, until the gaussians are no longer moving. Whereas in the estimation, the responsibility that each Gaussian has for each data point is calculated, during the maximization the mean of each Gaussian is moved towards the centroid of the entire data set.

The major problem with this approach, of course, is the if the data do not conform to the assumptions made by the technique then the latter may impose structure on the data and not disclose the 'true' structure.

Since the mixture of gaussians it is used in Chapter 5 in order to contrast the results of this method against the ones obtained with the method that we develop and present in this work, a further explanation of this method is presented.

Mixture of Gaussians

In practice, each cluster can be mathematically represented by a parametric distribution, like a Gaussian (continuous) or a Poisson (discrete). The entire data set is therefore modeled 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 traits:

- Component distributions have high 'peaks', i.e., data in one cluster are tight.
- The mixture model 'covers' the data well, i.e., dominant patterns in the data are captured by component distributions.

As a model-based approach, main advantages of this method are:

- Well-studied statistical inference techniques available.
- Flexibility in choosing the component distribution.
- Obtain a density estimation for each cluster.
- A 'soft' classification is available.

It is very important to notice that in this algorithm some assumptions have to be made:

1. The samples come from a known number K of classes.
2. Prior probabilities for each class, mixing parameters, are known:
$$P(\omega_j); \qquad j = 1, \dots, K$$
3. The form of the class-conditional probabilities densities are known:
$$f_{\boldsymbol{x}|\omega_j,\boldsymbol{\theta}_j}(\boldsymbol{x}|\omega_j, \boldsymbol{\theta}_j) \tag{2.22}$$
4. The values of the parameters are unknown: $\boldsymbol{\theta}_j$.
5. The category labels are unknown (unsupervised).

In addition to that, in a mixture model, one makes two assumptions of independence abput the underlying prior distribution of tha classificationand conditional distribution of the observations given the classification. What we really want to maximize is the mixture density:

$$f_{\boldsymbol{x}|\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{j=1}^{K} f_{\boldsymbol{x}|\omega_j,\boldsymbol{\theta}_j}(\boldsymbol{x}|\omega_j, \boldsymbol{\theta}_j)P(\omega_j) \tag{2.23}$$

since it is the base to write the likelihood function of the statistical independent observed samples:

$$\begin{aligned} f_{\boldsymbol{Z}} &= \prod_{i=1}^{N} f_{\boldsymbol{x}_i}(\boldsymbol{x}_i|\boldsymbol{\theta}) \\ &= \prod_{i=1}^{N}\sum_{j=1}^{K} f_{\boldsymbol{x}_i|\omega_j,\boldsymbol{\theta}_j}(\boldsymbol{x}_i|\omega_j, \boldsymbol{\theta}_j)P(\omega_j) \end{aligned} \tag{2.24}$$

being $\boldsymbol{Z} = (\boldsymbol{x}_1\ \boldsymbol{x}_2\ \dots\ \boldsymbol{x}_N)$.
Now we should maximize the likelihood function by calculating:

$$\nabla_{\boldsymbol{\theta}_j} L = 0, \qquad \text{where} \quad L = \sum_{i=1}^{N} \ln\Big(f_{\boldsymbol{x}_i|\boldsymbol{\theta}}\Big)$$

Assuming statistical independence between $\boldsymbol{\theta}_k$, $\boldsymbol{\theta}_j$, ML (Maximum Likelihood) solutions is one of the multiple solutions of:

$$\nabla_{\boldsymbol{\theta}_j} L = \sum_{i=1}^{N} P(\omega_j|\boldsymbol{x}_i, \hat{\boldsymbol{\theta}}) \nabla_{\theta_j} \ln\Big(f_{\boldsymbol{x}_i}(\boldsymbol{x}_i|\omega_j, \hat{\boldsymbol{\theta}}_j)\Big) = 0, \qquad j = 1, \dots, K \tag{2.25}$$

Generalizing to the unknown prior probability case, we must:

1. Compute prior probability estimates:

$$\hat{P}(\omega_j) = \frac{1}{N} \sum_{i=1}^{N} \hat{P}(\omega_j|\boldsymbol{x}_i, \hat{\boldsymbol{\theta}}_j) \tag{2.26}$$

2. Compute vector parameter estimates:

$$\sum_{i=1}^{N} \hat{P}(\omega_j|\boldsymbol{x}_i, \hat{\boldsymbol{\theta}}) \nabla_{\theta_j} \ln\Big(f_{\boldsymbol{x}_i}(\boldsymbol{x}_i|\omega_j, \hat{\boldsymbol{\theta}}_j)\Big) = 0, \qquad j = 1, \dots, K \tag{2.27}$$

3. Compute conditioned probability for classes:

$$\hat{P}(\omega_j|\boldsymbol{x}_i, \hat{\boldsymbol{\theta}}) = \frac{f_{\boldsymbol{x}_i}(\boldsymbol{x}_i|\omega_j, \hat{\boldsymbol{\theta}}_j)\hat{P}(\omega_j)}{\sum_{k=1}^{K} f_{\boldsymbol{x}_i}(\boldsymbol{x}_i|\omega_k, |\hat{\theta}_k)\hat{P}(\omega_k)} \tag{2.28}$$

This procedure would be difficult. That is the reason why we use a simplified algorithm called EM (Expectation-Maximization) in order to solve it. The EM algorithm has become a popular tool in statistical estimation problems involving incomplete data, or in problems which can be posed in a similar form, such as mixture estimation. Thus, the EM algorithm is an efficient iterative procedure to compute the Maximum Likelihood (ML) estimate in the presence of missing or hidden data. In ML estimation, we wish to estimate the model parameters for which the observed data are most likely.

Each iteration of the EM algorithm consists in two processes: the E-Step, and the M-Step. In the expectation the missing data are estimated given the observed data and current estimate of the model parameters. This is achieved using the conditional expectation, explaining the choice of terminology. In the maximization, the likelihood is maximized under the assumption that the missing data are known. the estimate of the missing data from the E-Step are used in *lieu* of the actual missing data.The convergence is assured since the algorithm is guaranteed to increase the likelihood in each iteration.

The most widely used clustering method of this kind is the one based on learning a mixture of Gaussians: we can actually consider clusters as Gaussian distributions centered on their barycenters, as we can see in Fig. 2.6, where the grey circle represents the first variance of the distribution:

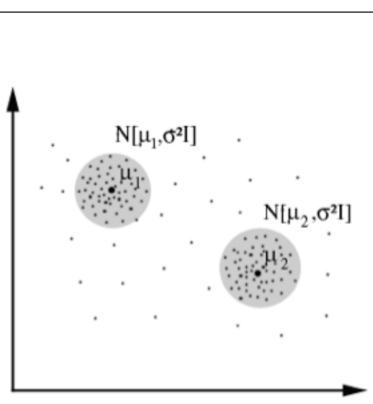

Figure 2.6: Mixture of Gaussians, 2 class and 2-D example

In this case

$$\ln\Big(f_{\boldsymbol{x}_i}(\boldsymbol{x}_i|\omega_j, \boldsymbol{\theta}_j) \Big) = \ln\left(\frac{|\boldsymbol{\Sigma}_j^{-1}|^{\frac{1}{2}}}{(2\pi)^{\frac{d}{2}}} \right) - \frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_j) \qquad (2.29)$$

and the parameters to estimate are:

$$\boldsymbol{\theta} = (\boldsymbol{\theta}_1 \ \boldsymbol{\theta}_2 \ \ldots \ \boldsymbol{\theta}_K) \qquad \boldsymbol{\theta}_j = (\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \qquad (2.30)$$

Lets now see how the EM algorithm works for a mixture of Gaussians, which produces a ‘soft’ assignment, combining Eq. 2.25 with Eq. 2.29, and knowing that the parameter to estimate are the ones shown in Eq. 2.30:

1. Initialize parameters: $j = 1, \ldots, K$

$$P(\omega_j) = \frac{1}{K} \qquad \boldsymbol{\theta} = (\boldsymbol{\theta}_1\ \boldsymbol{\theta}_2\ \ldots\ \boldsymbol{\theta}_K\ )$$

where $\boldsymbol{\theta}_j = (\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$,

$\boldsymbol{\mu}_j$: random sample point within the data $\qquad \boldsymbol{\Sigma}_j = \sum_{i=1}^{N} (\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T$

2. E-step:

$$\hat{P}r(\omega_j|\boldsymbol{x}_i, \hat{\boldsymbol{\theta}}) = \frac{|\hat{\boldsymbol{\Sigma}}_j^{-1}|^{\frac{1}{2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_j)^T \hat{\boldsymbol{\Sigma}}_j^{-1} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_j)\right\} P(\omega_j)}{\sum_{k=1}^{K} |\hat{\boldsymbol{\Sigma}}_k^{-1}|^{\frac{1}{2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)^T \hat{\boldsymbol{\Sigma}}_k^{-1} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)\right\} \hat{P}(\omega_k)} \tag{2.31}$$

3. M-step:

$$\begin{aligned}
\hat{P}(\omega_j) &= \frac{1}{N} \sum_{i=1}^{N} \hat{P}(\omega_j|\boldsymbol{x}_i, \hat{\boldsymbol{\theta}}) \\
\hat{\boldsymbol{\mu}}_j &= \frac{\sum_{i=1}^{N} \hat{P}(\omega_j|\boldsymbol{x}_i, \hat{\boldsymbol{\theta}})\boldsymbol{x}_i}{\sum_{i=1}^{N} \hat{P}(\omega_j|\boldsymbol{x}_i, \hat{\boldsymbol{\theta}})} \qquad (2.32) \\
\hat{\boldsymbol{\Sigma}}_j &= \frac{\sum_{i=1}^{N} \hat{P}(\omega_j|\boldsymbol{x}_i, \hat{\boldsymbol{\theta}})(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_j)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_j)^T}{\sum_{i=1}^{N} \hat{P}(\omega_j|\boldsymbol{x}_i, \hat{\boldsymbol{\theta}})}
\end{aligned}$$

4. Iterations Stop when the likelihood function:

$$\prod_{i=1}^{N} \sum_{j=1}^{K} f_{\boldsymbol{x}_i|\omega_j, \hat{\boldsymbol{\theta}}_j}(\boldsymbol{x}_i|\omega_j, \hat{\boldsymbol{\theta}}_j)\hat{P}(\omega_j)$$

does not vary.

We can repeat this procedure R times ($R = 10$ would be a good choice), what means starting with R different random initializations, and the parameters estimation which gives the highest value of the likelihood function is chosen.

Notice that, from the above equations, it can be inferred that from a distance measure point of view the measure used for compute the distance between the samples and the cluster centers is the Mahalanobis distance, Eq. 2.7

There are other methods as 'user-guided' or 'constraint-based', where the clustering is done by considering user-specified or application-specific constraint. As we will see in the next chapters, the method we develop is a partitioning method but a constraint-based one. In our case, the constraint, the pixel spatial location in the image, is introduced by a penalty term, that acts as a regularizer in the algorithm. More on that in the next Chapter.

The EM algorithm and K-means clustering
From another point of view, the K-means algorithm, explained before in subsection 2.1.3, can be considered as a widely used vector quantization procedure derived from the EM algorithm when $\boldsymbol{\Sigma}_j = \sigma^2 \boldsymbol{I}$. In this case $\boldsymbol{\theta}_j = \boldsymbol{\mu}_j$, and the EM update formula reduces to the K-means algorithm:

$$\hat{P}(\omega_j|\boldsymbol{x}_i, \hat{\boldsymbol{\mu}}) = \begin{cases} 1 & d(\boldsymbol{x}_i, \hat{\boldsymbol{\mu}}_j) < d(\boldsymbol{x}_i, \hat{\boldsymbol{\mu}}_k); j \neq k \\ 0 & other \end{cases} \tag{2.33}$$

$$\hat{P}(\omega_j) = \frac{1}{N} \sum_{i=1}^{N} \hat{P}(\omega_j|\boldsymbol{x}_i, \hat{\boldsymbol{\theta}}) = \frac{n_j}{N} \tag{2.34}$$

$$\hat{\boldsymbol{\mu}}_j = \frac{1}{n_j} \sum_{i=1}^{n_j} \boldsymbol{x}_i \tag{2.35}$$

The EM algorithm and fuzzy C-means clustering
As highlighted by Hathaway[26], the EM algorithm in the case of mixture models is formally equivalent to an alternative function that represents the fuzzy criterion. Therefore, the parameters u_{ik} (Eq. 2.20), with the corresponding manipulations can be computed by a formula equivalent to the one used in the E-step (Eq. 2.31). Similarly, the parameters $\boldsymbol{c}_k$ (Eq. 2.21) can be obtained as in the M-step (Eq. 2.32).

Since the calculations are the same in both methods, the EM algorithm applied to estimate the parameters of a mixture can be interpreted as the alternative optimization of the fuzzy criterion optimization.

There are other methods as Wavelet Analysis

2.2 Cluster Validity

The validation of clustering structures is the most difficult and frustrating part of cluster analysis. Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage.

Algorithms for Clustering Data, Jain and Dubes.

2.2.1 Cluster Evaluation

Every algorithm has its pros and cons, not only about cluster quality, but also about complexity, number of clusters in advance, etc. For what concerns cluster quality, we can evaluate, or better, validate, clusters.

In the case of supervised classification we have a variety of measures to evaluate how good our model is, e.g., accuracy, precision, see the definitions in section 1.3.3. For cluster analysis, the analogous question is: how can we evaluate the ‘goodness’ of the resulting clusters? But most of all: why should we evaluate it?

There are several reasons that answer this question. The main ones are: to determine the clustering tendency of the dataset, that is distinguish whether non-random structure actually exists in the data; to determine the correct number of clusters; to evaluate how well the results of a cluster analysis fit the data without reference to external information; to compare the results of a cluster analysis to externally known results, such as externally provided class labels; and to compare two sets of clusters to determine which is better. Notice that the first three are unsupervised techniques, while the last two require external info, and the last three can be applied to the entire clustering or just to individual clusters. Cluster evaluation has a number of challenges to be considered. A measure of cluster validity may be quite limited in the scope of its applicability (i.e. dimensions of the problem). We need a framework to interpret any measure (how good is '10'). And we have to take into consideration that if a measure is too complicated to apply or to understand, nobody will use it.

2.2.2 Estimation of the Number of Clusters

The selection of the value of K, number of clusters, depends on the goal. For example, it can be estimated from the data, examining the within-cluster dissimilarity W_k as a function of the number of clusters. Separate solutions are obtained for $k \in \{1, 2, \dots, K\}$, in fact $\{W_1, W_2, \dots, W_K\}$ decrease with increasing k. If $K < K^*$, where K^* is the optimum, means that each cluster returned contains a subset of the natural clusters, and the solution criterion value decreases substantially with each successive increase. Otherwise, if $K > K^*$ one of the estimated clusters must split at least one of the natural groups, which means a smaller decrease in the criterion. Thus, splitting a natural group within which the observations are all quite close to each other, reduces the criterion less than partitioning the union of two well-separated groups into their proper constituents. Therefore, there will be a sharp decrease in $K = K^*$.

$$\{W_K - W_{K-1} | K < K^*\} << \{W_K - W_{K-1} | K \geq K^*\} \tag{2.36}$$

An easy way to find K^* with that method is using the *Gap statistic* [25]: that consists on look for the number of clusters for which there is a knee, peak, or dip in the plot of the evaluation measure when it is plotted against the number of clusters. Of course, this is not always an easy task.

2.2.3 Measures of Cluster Validity

In cluster analysis, the numerical measures[19] that are applied to judge various aspects of cluster validity are classified into the following three types:

External (supervised) **Indices:** used to measure the extent to which cluster labels match externally supplied class labels (entropy).

Internal (unsupervised) **Indices:** used to measure the goodness of a clustering structure without respect to external information, as cluster cohesion vs cluster separation , i.e. Sum of Squared Error (SSE).

Relative Indices: used to compare two different clusterings or clusters (often an external or internal index is used for this function, e.g., SSE or entropy).

External Measures

- Entropy: is the degree to which each cluster consists of objects of a single class.
- Purity: is another measure of the extent to which a cluster contains objects of a single class.
- Precision: is the fraction of a cluster that consists of objects of a specified class.
- F-measure: is a combination of both, precision and recall, that measures the extent to which a cluster contains only objects of a particular class and all objects of that class.

Internal Measures

- Graph-based view: in Fig. 2.7 the left picture shows an example of cohesion graph-based evaluation, and the right one of separation graph-based evaluation.

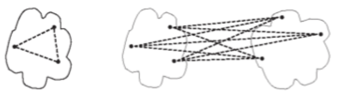

Figure 2.7: Graph-based clustering evaluation

- Prototype-based view: as in the graph-based method, the left and right pictures of Fig. **??** are cohesion and separation, respectively, examples of prototype-based clustering evaluation.

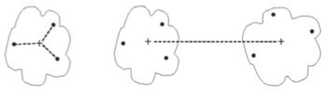

Figure 2.8: Prototype-based clustering evaluation

- Cluster Cohesion: measures how closely related are objects in a cluster. Cohesion is measured by the within cluster sum of squares (SSE):

$$WSS = \sum_{j=1}^{K} \sum_{\boldsymbol{x}_i \in \omega_j} (\boldsymbol{x}_i - \boldsymbol{\mu}_j)^2 \qquad (2.37)$$

- Cluster Separation: measures how distinct or well-separated a cluster is from other clusters. Separation is measured by the between cluster sum of squares:

$$BSS = \sum_{j=1}^{K} |K_j|(\boldsymbol{\mu} - \boldsymbol{\mu}_j)^2 \qquad (2.38)$$

where $\boldsymbol{\mu}_j$and $|K_j|$ are the centroid (barycenter), and the size of cluster ω_j, respectively. And $\boldsymbol{\mu}$ is the mean of the clusters' centers.

Consequently, the Squared Sum Error can be defined as:

$$SSE = WSS + BSS \tag{2.39}$$

So far, we have focused on evaluation of a group of clusters. Many of these measures, however, also can be used to evaluate individual clusters and objects. For example, a cluster with a high cohesion may be considered better than a cluster with a lower one.

This information often can be used to improve the quality of the clustering. For instance, split not very cohesive clusters or merge not very separated ones. We can also evaluate the objects within a cluster in terms of their contribution to the overall cohesion or separation of the cluster.

- Silhouette Coefficient: combines ideas of both cohesion and separation, but for individual points, as well as clusters and clusterings. For an individual point, $\boldsymbol{x}_i$:
  - Calculate a_i = average distance of $\boldsymbol{x}_i$ to the points in its cluster.
  - Calculate b_i = min(average distance of i to points in another cluster).
  - The silhouette coefficient for a point is then given by:

$$s_i = (b_i - a_i)/\max(a_i, b_i) \tag{2.40}$$

- Correlation: if we are given the similarity matrix for a data set and the cluster labels from a cluster analysis of the data set, then we can evaluate the "goodness" of the clustering by looking at the correlation between the similarity matrix and an ideal version of the similarity matrix based on the cluster labels. An ideal matrix consists on one row and one column for each data point, where an entry is 1 if the associated pair of points belong to the same cluster, and an entry is 0 if the associated pair of points belongs to different clusters.
  To evaluate the correlation the correlation between the two matrices has to be computed (since the matrices are symmetric, only the correlation between $N(N-1)/2$ entries needs to be calculated). A high correlation indicates that points that belong to the same cluster are close to each other.

2.2.4 Framework for Cluster Validity

As important as any measure is to have a framework to interpret them. Statistics provide a framework for cluster validity.

The more atypical a clustering result is, the more likely it represents valid structure in the data. In those circumstances, we can compare the values of an index that result from random data or clusterings to those of a clustering result, if the value of the index is unlikely, then the cluster results are valid. These approaches are more complicated and harder to understand. In fact, for comparing the results of two different sets of cluster analyses, a framework is less necessary. However, there is the question of whether the difference between two index values is significant.

Chapter 3

Previous Work: Spatial Fuzzy Clustering using Markov Random Fields

3.1 Spatial Clustering

In exploratory data analysis, clustering techniques aim to summarize a set of objects by grouping together similar objects in the same class. In this work, we are interested in applications where the objects are both described by variables and located at neighboring geographic sites, and where the two following goals are aimed when partitioning the observations into clusters (the number of which are supposed to be known):

1. The clusters should be as homogeneous as possible, i.e. observations should be as similar as possible within a group;
2. It is assumed that the partition changes slowly in the geographic space, i.e. two neighboring observations are more likely to belong to the same group than two observations lying far apart.

Such clustering problems arise in various applications. One example is in ecology, when contiguous quadrates have to be partitioned according to presence/absence measures of animal species. Another example is unsupervised image segmentation, the one we are concerned about, where the sites are the pixels of the image and the variables are the observed intensities in spectral bands. As we will see in the next Chapter, were we explain how our images are, this intensities can represent different kind of image characteristics/features.

3.2 Spatial Fuzzy Clustering

3.2.1 Introduction

In many real situations the conventional hard clustering, which restrict each point of the data set to exclusively just one cluster, make segmentation a difficult task because of issues such as limited spatial resolution, poor contrast, overlapping intensities, noise and intensity inhomogeneities. Due to the fuzzy idea of partial membership of belonging described by a fuzzy function, set the fuzzy clustering as a soft segmentation method. Among the fuzzy clustering methods, fuzzy C-means is the most popular in image segmentation because it has robust characteristics for ambiguity and can retain much more information than hard segmentation methods. However, this method has a serious limitation, i.e., it does not incorporate any information about spatial context.

Since the aim of this work is the study of spatial clustering, i.e. discover interesting spatial patterns and features given a data base, we learnt about Spatial Fuzzy Clustering (SFC), which happen to be very appropriate to capture intrinsic relationships between spatial and non-spatial data. It is interesting to know that the SFC framework can be adapted to any known general fuzzy clustering method, and it happens that spatial fuzzy C-means (FCM) is a special case of the more general framework.

Inasmuch as the fuzzy clustering criterion (subsection 2.1.3, page 31), the mixture model based approach to clustering analysis and the estimation of mixture parameters by the EM algorithm (subsection 2.1.7, page 37), which it is very important and useful in image segmentation (we will talk about it in the next sections); as well as, the equivalence between this estimation procedure and the optimization of the fuzzy criterion, are widely described in the previous Chapter, we thought that it is not necessary to insist on explaining them again.

3.2.2 Previous Algorithms

To compensate the drawback of FCM algorithm, i.e. the lack of spatial information, the obvious way is to smooth the image before segmentation. However, the conventional smoothing filters can result in loss of important image details, especially boundaries or edges of image. More importantly, there is no way to rigorously control the trade-off between smoothing and clustering.

Hence, other different approaches have been proposed. In Ref.[54], Tolias proposed a fuzzy ruled-based scheme called the rule-based neighborhood enhancement system to impose spatial continuity by post-processing on the clustering results obtained by FCM algorithm. Noordam[46] proposed a geometrically guided FCM algorithm based on a semi-supervised FCM technique for multivariate image segmentation; in their work, the geometrical condition information of each pixel is determined by taking into account the local neighborhood of each pixel.

Recently, approaches by directly modifying the objective functions have been proposed to increase the robustness of FCM when applied to spatial segmentation. The algorithm is formulated by incorporating the spatial neighborhood information into the original FCM algorithm with a penalty term, that acts as a regularizer in the algorithm. In Ref.[36] a new dissimilarity index that considers the influence of the neighbor pixels on the centre pixel was presented to replace the conventional normed distance in the FCM algorithm; however, this method can handle only a small amount of noise. In Ref.[1] a regularization term was introduced into the standard FCM to impose neighborhood effect. Later, Ref.[35], this regularization term was incorporated into the Adaptive FCM (AFCM); although this method is promising, it is computationally expensive that means more consuming time is needed during the computation.

As we mentioned before, the Neighborhood EM (NEM) algorithm is very used in image segmentation, and it is highly related to the fuzzy spatial clustering. In fact, some of the fuzzy clustering methods are inspired by the NEM algorithm. Therefore, NEM is slightly explained below.

Neighborhood EM algorithm

In order to incorporate the spatial dependence into the objects, a modified version of the conventional expectation maximization (EM) algorithm has been proposed in[3]. In this approach, the maximum likelihood criterion is penalized by a term that quantifies the degree of spatial contiguity of the pixels supporting the respective components of the probability density function (pdf) model. The spatial structure of a given data set is defined by using a matrix $\boldsymbol{W} = (w_{ij})$, where

$$w_{ij} = \begin{cases} 1 & if\ \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are neighbors and } i \neq j \\ 0 & otherwise \end{cases} \tag{3.1}$$

The following term is then used for regularizing the maximum likelihood criterion:

$$G(K) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{K} \upsilon_{ki} \upsilon_{kj} w_{ij}$$

where K is the number of classes and υ_{ki} is the probability that $\boldsymbol{x}_i$ belongs to class k. This term characterizes the level of homogeneity of the partition. The more the classes contain adjacent elements, the greater the term is. In this case, the NEM criterion is obtained by optimizing the weighted sum of two terms:

$$U(K, \boldsymbol{\theta}) = D(K, \boldsymbol{\theta}) + \beta G(K)$$

where $D(K, \boldsymbol{\theta})$ is the log-likelihood function of EM algorithm, and $\beta > 0$ is a fixed coefficient. Details about NEM can be found in Ref.[3]. This algorithm is maximized to get the optimum results just as the same structure as the EM algorithm. Successful results have been reported for image segmentation using this methods.

It is important to notice that the clustering result of the NEM algorithm depends largely on the choice of the spatial coefficient β. As we will see later, in Chapter 5, the method proposed in this work is also largely dependent on this regularization parameter. Thus, in that Chapter there is a further discussion about how to determine this parameter.

Penalized FCM Algorithm

Inspired by this algorithm (NEM), some algorithms have been developed. One example is the Penalized FCM (PFCM) algorithm, presented in[55], in which the penalty term takes the spatial dependence of the objects into consideration basing on NEM algorithm and modifying it according to the FCM criterion. The PFCM algorithm is then proposed by minimizing this new objective function according to the zero gradient condition, which can handle both the feature space information and the spatial information during segmentation. In addition, in this algorithm the membership is changed while the centroid computations are the same as in the standard FCM method.

$$J_{PFCM} = \sum_{i=1}^{N}\sum_{k=1}^{K}(u_{ki})^q d^2(\boldsymbol{x}_i, c_k) + \gamma \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{K}(u_{ki})^q(1-u_{kj})^q w_{ij} \qquad (3.2)$$

where u_{ki} is the degree of membership, $\boldsymbol{c}_k$ are the centroid corresponding to class k and w_{ij} is defined as in Eq. 3.1. The γ regularization parameter is inversely proportional to the signal-to-noise ratio (SNR).

The major difference between NEM and PFCM algorithms is that the penalty term in the NEM is maximized to get the solutions, while in the PFCM it should be minimized in order to satisfy the principles of FCM. Besides, the penalty term in the PFCM has a weighting exponent to control the degree of fuzziness in the resulting membership function contrary to the penalty term in the NEM algorithm that is crisp.

This penalty term is minimized when the membership value for a particular class is large and the membership values for the same class at neighboring pixels is also large, and vice versa. In other words, it constrains the pixel's membership value of a class to be correlated with those neighboring pixels. An iterative algorithm for minimizing Eq. 3.2 can be derived by evaluating the centroids and membership functions that satisfy a zero gradient condition. The constrained optimization in Eq. 3.2 will be solved using one Lagrange multiplier.

Relation with MRF

All the above methods are claimed to be similar to the Markov random field theory, but they are not directly based on MRF, the very efficient and competent theory to describe spatial context information in image analysis.

Recently, some approaches which apply similar reasoning as the above methods for the Markov random field based approach to spatial clustering, have been developed. The *prior* spatial information is introduced through the Markov Random field theory and its equivalent Gibbs random field theory, in which the spatial information is encoded through mutual influences of neighboring sites. Therefore, in the following section we present the Markov random field theory and its applications in spatial clustering.

3.3 Markov Random Field and Fuzzy Spatial Clustering

When only the first goal of spatial clustering is in order, i.e. when the spatial information is not used, mixture models and the Expectation-Maximization (EM) algorithm[14] may provide a relevant approach. Recalling Hathaways result[26], which established that applying the EM algorithm to estimation for the parameters of a mixture is formally equivalent to the alternative gradient optimization of a fuzzy clustering criterion (that equivalence is explained in subsection 2.1.7, page 41); this criterion contains a 'fuzzy' sum of within-cluster inertia.

When both goals are considered, i.e. when a hypothesis of spatial smoothness of the partition has to be accounted for, this note points out the relevance of EM and Markov random fields. Applying the principle of EM to a hidden Markov random field model may still be interpreted as the alternative optimization of another fuzzy clustering criterion; this new criterion contains on the one hand the fuzzy sum of within-cluster inertia, and on the other hand a spatial smoothing function; the equivalence is based on the algorithms proposed by Zhang (Zhang 1992) and Ambroise[3].

3.3.1 MRF Introduction

Several methods are investigated in order to partition in groups a set of multivariate observation vectors located at neighboring geographic sites; applications include image segmentation. In this perspective, the deterministic variant of the EM procedure described in Zhang (1992) for hidden Markov random fields is shown to be equivalent to the optimization of a spatial fuzzy clustering criterion using the so-called Neighborhood EM algorithm[3], explained in the previous section (3.2.2). The obtained fuzzy partition can be interpreted as the posterior probabilities that the observations belong to the K groups, computed by an efficient iterative method based on the mean field approximation principle. The resulting algorithm may be viewed as an extension of the K-means algorithm to fuzzy clustering and spatial data.

The Markov Random Field method (MRF) is considered a powerful stochastic tool to model the joint probability distribution of the image pixels in terms of local spatial interactions[6][21][34]. MRF models can be used not only to extract texture features from image textures but also to model the image segmentation problem, as from the viewpoint of the random field a segmentation result is a label distribution in the same lattice as the original image.

In real scenes, neighboring pixels usually have similar intensities. In a probabilistic framework, such regularities are well expressed by Markov Random Fields. On the other hand, the local behavior of MRF permits to develop highly parallel algorithms.

Using MRF models for image segmentation has a number of advantages. First, the spatial relationship can be seamlessly integrated into a segmentation procedure. Second, the MRF based segmentation model can be inferred in the Bayesian framework which is able to utilize various kinds of image features. Third, the label distribution can be obtained when maximizing the probability of the MRF model.

3.3.2 Notation and definition

From[34] :
Let $\boldsymbol{F} = \{\boldsymbol{F}_1, \ldots, \boldsymbol{F}_N\}$ be a family of random variables defined on the set $\boldsymbol{S}$, in which each random variable takes a value $\boldsymbol{f}_i$ in $\boldsymbol{\Lambda}$.

The family $\boldsymbol{F}$ is called a random field.
We use the notation $\boldsymbol{F}_i = \boldsymbol{f}_i$ to denote the event that $\boldsymbol{F}_i$ takes the value $\boldsymbol{f}_i$ and the notation

$$(\boldsymbol{F}_1 = \boldsymbol{f}_1, \ldots, \boldsymbol{F}_N = f_N)$$

to denote the joint event. For simplicity, a joint event is abbreviated as $\boldsymbol{F} = f$ where $\boldsymbol{f} = \{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\}$ is a configuration of $\boldsymbol{F}$, corresponding to a realization of the field.

For a discrete label set $\boldsymbol{\Lambda}$, the probability that random variable $\boldsymbol{F}_i$ takes the value $\boldsymbol{f}_i$ is denoted $P(\boldsymbol{F}_i = \boldsymbol{f}_i)$, abbreviated $P(\boldsymbol{f}_i)$ unless there is a need to elaborate the expressions, and the joint probability is denoted

$$P(\boldsymbol{F} = \boldsymbol{f}) = P(\boldsymbol{F}_1 = \boldsymbol{f}_1, \ldots, \boldsymbol{F}_N = \boldsymbol{f}_N)$$

and abbreviated $P(\boldsymbol{f})$.

For a continuous $\boldsymbol{\Lambda}$, we have probability density functions (p.d.f.'s), $p(\boldsymbol{F}_i = \boldsymbol{f}_i)$ and $p(\boldsymbol{F} = \boldsymbol{f})$.

$\boldsymbol{F}$ is said to be a Markov random field on $\boldsymbol{S}$ with respect to a neighborhood system Q if and only if the following two conditions are satisfied:

$$P(\boldsymbol{f}) > 0, \qquad \boldsymbol{f} \in \boldsymbol{F} \qquad \text{(positivity)} \qquad (3.3)$$

$$P(\boldsymbol{f}_i | \boldsymbol{f}_{\boldsymbol{S}-\{i\}}) = P(\boldsymbol{f}_{|Q_i}) \qquad \text{(Markovianity)} \qquad (3.4)$$

where $\boldsymbol{S} - \{i\}$ is the set difference, $\boldsymbol{f}_{\boldsymbol{S}-\{i\}}$ denotes the set of labels at the sites in $\boldsymbol{S} - \{i\}$ and

$$\boldsymbol{f}_{Q_i} = \{\boldsymbol{f}_j | j \in Q_i\} \qquad (3.5)$$

stands for the set of labels at the sites neighboring i.

The positivity is assumed for some technical reasons and can usually be satisfied in practice. For example, when the positivity condition is satisfied, the joint probability $P(\boldsymbol{F})$ of any random field is uniquely determined by its local conditional probabilities (Besag[6]).

The Markovianity depicts the local characteristics of $\boldsymbol{F}$. A label interacts with only the neighboring labels. In other words, only neighboring labels have direct interactions with each other. It is always possible to select sufficiently large Q_i so that the Markovianity holds. The largest neighborhood consists of all other sites.

Any $\boldsymbol{F}$ is an MRF with respect to such a neighborhood system.

An MRF can have other properties such as homogeneity and isotropy. It is said to be homogeneous if $P(\boldsymbol{f}_i | \boldsymbol{f}_{Q_i})$ is regardless of the relative position of site i in $\boldsymbol{S}$.The isotropy will be illustrated later with clique potentials.

It may need to define for some problem a few coupled MRFs, each defined on one of the spatially interwoven sets of sites. For example, in the related tasks of image restoration and edge detection, two MRFs, one for pixel values ($\{\boldsymbol{f}_i\}$) and the other for edge ($\{\boldsymbol{l}_{i,j}\}$), can be defined on the image lattice and its dual lattice, respectively. They are coupled to each other e.g. via conditional probability $P(\boldsymbol{f}_i | \boldsymbol{f}_j, \boldsymbol{l}i, j)$.

Markov Process

The concept of MRFs is a generalization of that of Markov processes (MPs) which are widely used in sequence analysis. An MP is defined on a domain of time rather than space. It is a sequence (chain) of random variables

$$\{\ldots, \boldsymbol{F}_1, \ldots, \boldsymbol{F}_N, \ldots\}$$

defined on the time indices $\{\ldots, 1, \ldots, N, \ldots\}$. An nth order unilateral MP satisfies:

$$P(\boldsymbol{f}_i | \ldots, \boldsymbol{f}_{i-2}, \boldsymbol{f}_{i-1}) = P(\boldsymbol{f}_i | f_{i-1}, \ldots, \boldsymbol{f}_{i-n}) \tag{3.6}$$

A bilateral or non-causal MP depends not only on the past but also on the future. An n-th order bilateral MP satisfies:

$$P(\boldsymbol{f}_i | \ldots, \boldsymbol{f}_{i-2}, \boldsymbol{f}_{i-1}, \boldsymbol{f}_{i+1}, \boldsymbol{f}_{i+2}, \ldots) = P(\boldsymbol{f}_i | f_{i+n}, \ldots, \boldsymbol{f}_{i+1}, \boldsymbol{f}_{i-1}, \ldots, \boldsymbol{f}_{i-n}) \tag{3.7}$$

It is generalized into MRFs when the time indices are considered as spatial indices.

There are two approaches for specifying an MRF, that in terms of the conditional probabilities $P(\boldsymbol{f}_i | \boldsymbol{f}_{Q_i})$ and that in terms of the joint probability $P(\boldsymbol{f})$.

Besag[6] argues for the joint probability approach in view of the disadvantages of the conditional probability approach. Firstly, no obvious method is available for deducing the joint probability from the associated conditional probabilities. Secondly, the conditional probabilities themselves are subject to some non-obvious and highly restrictive consistency conditions. Thirdly, the natural specification of an equilibrium of statistical process is in terms of the joint probability rather than the conditional distribution of the variables. Fortunately, a theoretical result about the equivalence between Markov random fields and Gibbs distribution (Hammersley and Clifford[23] ; Besag[6]) provides a mathematically tractable means of specifying the joint probability of an MRF.

3.3.3 Gibbs Random Field Theory

A set of random variables $\boldsymbol{F}$ is said to be a Gibbs random field (GRF) on $\boldsymbol{S}$ with respect to Q if and only if its configurations obey a Gibbs distribution. A Gibbs distribution takes the following form:

$$P(\boldsymbol{x}) = \frac{1}{Z} exp\,(-U(\boldsymbol{f})) \tag{3.8}$$

According to the Hammersley-Clifford theorem, an MRF can be equivalently characterized by a Gibbs distribution.

Theorem 1 *(Hammersley-Clifford). $\boldsymbol{F}$ is a MRF with respect to the neighborhood system Q if and only if $\pi(\boldsymbol{f}) = P(\boldsymbol{F} = \boldsymbol{f})$ is a Gibbs distribution:*

$$\pi(\boldsymbol{f}) = \frac{1}{Z} \exp\Big(-U(\boldsymbol{f})\Big) = \frac{1}{Z} \exp\left(-\sum_{\boldsymbol{Q} \in \nu} V_C(\boldsymbol{f})\right)$$

where, ν is the set of all the possible cliques Z is the normalizing constant or partition function:

$$Z = \sum_{\boldsymbol{f} \in \boldsymbol{F}} \exp\Big(-U(\boldsymbol{f})\Big)$$

and $U(\boldsymbol{f})$ is the energy function and V_C denotes the clique[1] *potentials.*

[1]A clique C for $(\boldsymbol{S}, Q)$ is defined as a subset of sites in $\boldsymbol{S}$. It consists either of a single site $C = \{i\}$, or of a pair of neighboring sites $C = \{i, i'\}$, or of a triple of neighboring sites

3.3.4 Markov-Gibbs Equivalence

The equivalence between MRF and Gibbs distribution provides a simple way to specify MRF's through clique-potentials instead of local characteristics, which is usually very difficult.

An MRF is characterized by its local property (the Markovianity) whereas a GRF is characterized by its global property (the Gibbs distribution). As we have seen, the Hammersley-Clifford theorem establishes the equivalence of these two types of properties.

The theorem states that $\boldsymbol{F}$ is an MRF on $\boldsymbol{S}$ with respect to Q if and only if $\boldsymbol{F}$ is a GRF on $\boldsymbol{S}$ with respect to Q. Many proofs of the theorem exist, e.g. in (Besag[6]; Moussouris[43]; Kindermann and Snell[31]).

A proof that a GRF is an MRF is given in[34]. We do not include this demonstration because it is out of our objectives to go into such a mathematic theory, and it is not necessary in order to understand the method we develop in the following Chapter. Furthermore, that demonstration only proves that a Gibbs random field is a Markov random field. The proof that an MRF is a GRF is much more involved.

In what we are interested is the practical value of the theorem, the fact that it provides a simple way of specifying the joint probability. One can specify the joint probability $P(\boldsymbol{F} = \boldsymbol{f})$ by specifying the clique potential functions $V_C(\boldsymbol{f})$ and chooses appropriate potential functions for desired system behavior. In this way, it encodes the *a priori* knowledge or preference about interactions between labels.

How to choose the forms and parameters of the potential functions for a proper encoding of constraints is a major topic in MRF modeling. The forms of the potential functions determine the form of the Gibbs distribution. When all the parameters involved in the potential functions are specified, the Gibbs distribution is completely defined.

To calculate the joint probability of an MRF, which is a Gibbs distribution, it is necessary to evaluate the partition function. Because it is the sum over a combinatorial number of configurations in $\boldsymbol{F}$, the computation is usually intractable. The explicit evaluation can be avoided in maximum-probability based MRF vision models when $U(\boldsymbol{f})$ contains no unknown parameters, as we will see subsequently. However, this is not true when the parameter estimation is also a part of the problem. In the latter case, the energy function $U(\boldsymbol{f}) = U(\boldsymbol{f}|\boldsymbol{\theta})$ is also a function of parameters $\boldsymbol{\theta}$ and so is the partition function $\boldsymbol{\gamma} = \boldsymbol{\gamma}(\boldsymbol{\theta})$. The evaluation of $\boldsymbol{\gamma}(\boldsymbol{\theta})$ is required. To circumvent the formidable difficulty therein,

$C = \{i, i', i''\}$, and so on. The collections of single-site, pair-site and triple-site cliques will be denoted by:

$$C_1 = \{i|i \in \boldsymbol{S}\},\ C_2 = \{\{i, i'\}|i' \in Q_i\ i \in \boldsymbol{S}\} \text{ and}$$
$$C_3 = \{\{i, i', i''\}|i, i', i'' \in \boldsymbol{S} \text{ are neighbours to one another}\}, \text{ respectively.}$$

Note that the sites in a clique are ordered, and $\{i, i'\}$ is not the same clique as $\{i', i\}$, and so on. The collection of all cliques for $\boldsymbol{S}, Q$ is $C = C_1 \cup C_2 \cup C_3 \cup \ldots$ where '...' denotes possible sets of larger cliques.

the joint probability is often approximated in practice. Several approximate formulae have been developed in order to solve the problem of MRF parameter estimation.

3.3.5 Example: Multi-Level Logistic Model (MLL)

As we said in the preceding subsection, by choosing different clique potential function V_C , a wide variety of distributions can be formulated as Gibbs distributions. Here we introduce one of the most common models that uses the GRF-MRF theory, the Multi-Level Logistic Model (MLL).

In MLL models[20][34], the potentials for cliques containing more than one site are defined as:

$$V_C\boldsymbol{x} = \begin{cases} -\beta_C & if \text{ all sites in } C \text{ have the same label} \\ \beta_C & otherwise \end{cases} \tag{3.9}$$

$\beta_C > 0$ is a constant depending on the type of clique. This encourages neighboring sites to have the same class label. Homogeneity is imposed on the model by assigning the same potential function to all cliques of a certain type, independent of their positions in the image.

Other well-known methods exist, e.g. the Smoothness Prior MRF (Poggio[47]; Bertero[4]), and the Hierarchical GRF model (Derin and Cole[16]; Derin and Elliot[17]).

In fact, in the algorithm that we develop, a different clique potential function from the above is chosen. The energy term in our case is the Gaussian MRF model which is widely used in many applications. The Eq. 4.10, in Chapter 4, shows the Gaussian-MRF energy term, and the corresponding clique potential function.

3.3.6 Bayes Labeling of MRF

In the previous subsection we have introduced an example of clique potential function, but this function has to work as a penalty term, introducing spatial information, in a objective function that has to be optimized, in our case i order to get a suitable image segmentation. In the next paragraphs we explain how this objective function is formulated.

Research Issues

There are three basic issues in optimization-based vision: problem representation, objective function and optimization algorithms. There are two aspects of a representation: descriptive and computational. The former concerns how to represent image features and object shapes, which relates to photometry and geometry (Koenderink[32] ; Mundy and Zisserman[44]; Kanatani[29]) and is not an emphasis of this work. The later concerns how to represent the solution, which relates to the choice of sites and label set for a labeling problem. For example, in image segmentation, we may use a chain of boundary locations to represent the solution; we may alternatively use a region map to do the same job. Comparatively speaking, however, the region map is a more natural representation for MRFs.

The second issue is how to formulate an objective function for the optimization. The objective function maps a solution to a real number measuring the quality of the solution in terms of some goodness or cost. The formulation determines how various constraints, which may be pixel properties like intensity and color and/or context like relations between pixels or object features, are encoded into the function. Because the optimal solution which is the optimum of the objective function, the formulation defines the optimal solution.

The third is how to optimize the objective, i.e. how to search for the optimal solution in the admissible space. Two major concerns are (1) the problem of local minima existing in non-convex functions and (2) the efficiency of algorithms in space and time. They are somewhat contradictory and currently there is no algorithms which guarantee the global solution with good efficiency. These three issues are related to one another. In the first place, the scheme of representation influences the formulation of the objective function and the design of the search algorithm. On the other hand, the formulation of an objective function affects the search. For example, suppose two objective functions has the same point as the unique global optimum but one of them is convex whereas the other is not; obviously the convex one is much more desired because it provides convenience for the search.

Unfortunately, MRF based segmentation model is very easily trapped in local optima due to the imposed spatial homogeneity constraint imposed by the region labeling component. As a result, the feature modeling component might not be able to learn the global parameters.

As we will see in Chapter 4, we choose the Conjugate Gradient Method (CGM) for the numerical optimization of the objective function. Though it does not guarantee a global optima, this nonlinear method works in a effective way, and it is not time consuming. But more about this later, in that Chapter we will explain how CGM works .

Optimality Criteria

In formal models, as opposed to heuristic ones, an energy function is formulated based on an established criterion. Because of inevitable uncertainties in vision processes, principles from statistics, probability and information theory are often used as the formal basis.

When we have the knowledge about the data distribution but no appreciable prior information about the quantity being estimated, we may use the maximum likelihood (ML) criterion. When the situation is the opposite, that is, when we have only prior information, then we may use the maximum entropy criterion. Distributions of higher entropy are more likely because nature can generate them in more ways and the maximum entropy criterion is simply taking this fact into account (Jaynes[28]).

Neither of these two methods is adequate for problems where we know both prior and likelihood distributions. With both sources of information available, the best we can get is that maximizes a Bayes criterion. There are two forms of such estimate often used in practice: that of the MAP probability and that of maximum a posteriori mean. The maximizer of the posterior marginals (MPM) (Marroquin[40]; Marroquin et al.[41]) provides an alternative Bayes estimator.

Although there have been philosophical and scientific controversies about their appropriateness in inference and decision making (see Clark and Yuille[12] for a short review), Bayes criteria are among the most popular ones in computer vision and in fact, MAP is the most popular criterion in optimization-based MRF modeling.

The equivalence theorem of between Markov random fields and Gibbs distribution established in subsection 3.3.4 provides a convenient way for specifying the joint prior probability, solving a difficult issue in MAP-MRF labeling.

MAP-MRF Approach

The procedure of the MAP-MRF approach for solving computer vision problems is summarized in the following:

1. Pose a vision problem as one of labeling in categories $\{1, \ldots, K\}$ and choose an appropriate MRF representation $\boldsymbol{f}$.
2. Derive the posterior energy to define the MAP solution to a problem.
3. Find the MAP solution.

The process of deriving the posterior energy is summarized in[34].

1. Define a neighborhood system Q on $\boldsymbol{S}$ and the set of cliques C for Q.
2. Define the prior clique potentials $V_C(\boldsymbol{f})$ to give $U(\boldsymbol{f})$.
3. Derive the likelihood energy.

The prior model depends on the type of the scene (e.g. the type of surfaces) we expect. In vision, it is often one of the Gibbs models introduced in subsection 3.3.3. The likelihood model depends on physical considerations such as the sensor process (transformations, noise, etc.). It is often assumed to be Gaussian. The parameters in both models need be specified for the definitions of the models to be complete. The specifications can be something of arts when done manually and it is desirable that it is done automatically.

Summarizing, in a MAP-MRF problem we are concerned with the following issues:

1. Choosing an appropriate representation for the MRF labeling.
2. Deriving the a posteriori distribution of the MRF as the criterion function of the labeling solution. It mainly concerns the specification of the forms of the prior and the likelihood distributions. The involved parameters may or may not be specified at this stage.
3. Estimating involved parameters in the prior and the likelihood distributions. The estimation is also based on some criterion, very often, maximum likelihood. In the unsupervised case, it is performed together with MAP labeling.
4. Searching for the MRF configuration to maximize the posterior distribution. This is mainly algorithmic. The main issues are the quality (globality) of the solution and the efficiency.

3.3.7 Drawback

This problem is reduced to the optimization of a non-convex energy function, thus many local optima. As we said before, MRF based segmentation model is very easily trapped in local optima due to the imposed spatial homogeneity constraint imposed by the region labeling component. As a result, the feature modeling component might not be able to learn the global parameters.

Stewart[51] analyzed the relationship between the two terms (one the feature information, and the other one providing the spatial information) in their MRF model in detail and proposed a specific solution for the weighting parameter (they called it shape parameter) according to *a priori* information of the size of region shapes.

However, in our work we try to deal with that drawback running the algorithm with different initializations (random or fixed). The idea of running several times the algorithm is very common in methods that can fall in a local optima, e.g. K-means method.

3.3.8 In our work

Notation

In the next Chapter we introduce our method. It is important to notice that the notation, used to explain the theory we base on in order to develop it, is quite different from the one used in the previous section in order to explain the MRF theory.

- The $\boldsymbol{F}$ random field becomes the set of multispectral images $\boldsymbol{Y}$, where $\boldsymbol{Y} = \boldsymbol{y}$ is a realization of $\boldsymbol{Y}$, and $\boldsymbol{y}_i$ is the feature vector of the pixel at location i.

- The segmentation we want to obtain is the realization $\boldsymbol{\gamma} = \boldsymbol{\alpha}$, of the random field $\boldsymbol{\gamma}$. And $\boldsymbol{\alpha}_i$ is a vector representing the degree of membership of a pixel for each class.

Otherwise, this is widely explained in the following Chapter.

Optimization Procedure

Being $\boldsymbol{Y} = \{\boldsymbol{y}_i\}_{i \in \boldsymbol{S}}$ a set of image data ($\boldsymbol{y}_i$ is the value at pixel i).
The MAP estimation procedure through we want to find the labeling $\hat{\boldsymbol{\alpha}} \in \boldsymbol{\gamma}$ is the one which maximizes:

$$P(\gamma|\boldsymbol{Y}) \propto P(\boldsymbol{Y}|\boldsymbol{\alpha})P(\boldsymbol{\alpha})$$

where $\boldsymbol{\gamma}$ is the set of all possible global labelings, in our case the segmentation.

We also establish the following *Hypothesis*: $P(\boldsymbol{Y}|\boldsymbol{\alpha})$ is Gaussian and $P(\boldsymbol{\alpha})$ is Markovian.

To construct our final cost function, we need to choose the penalty term, based on the Gibbs theory in order to take into account the spatial information. The energy term corresponding to the *a priori* model described in the previous section is: $P(\boldsymbol{\alpha}) = \frac{1}{Z} \exp -U(\boldsymbol{\alpha})$, and in our case, selecting the Gaussian MRF model which is widely used in many applications for the energy function:

$$U(\boldsymbol{\alpha}) = \beta \sum_{i=1}^{N} \sum_{k=i}^{K-1} \sum_{j \in V_i} \left(|\alpha_{ik} - \alpha_{jk}|^2 \right)$$

where V_i represents the pixels belong to the i's neighborhood.

Eventually, the numerical optimization is performed for the nonlinear Conjugate Gradient Method.

3.4 Relationships between Spatial Clustering Methods

Model	**Known Parameters**	**Estimation EM**	**Fuzzy Clustering**	**Hard Clustering**
Mixture	Bayesian classifier	Dempster et. al.(1977)	Hathaway (1986)	K-means, CEM
Markov Random Field	MAP (Geman & Geman,1984) ICM (Besag, 1986)	Gibbsian EM, Mean Field (Zhang, 1992)	NEM (Ambroise, 1996)	ICM+estim. (Besag, 1986), NCEM (Ambroise, 1996)

Table 3.1: Relationships between clustering methods

To end this Chapter, it is a good idea to show the relationship between some clustering methods. Table 3.1[13] shows the links between discrimination, EM procedures, fuzzy and hard clustering methods, both in the non-spatial independence setup (first row) and in the spatial dependence setup (second row).

First column groups together classification methods based on a probabilistic model when all parameters are known. Second column lists the EM algorithms proposed to estimate the parameters of those models. Third column shows the associated fuzzy clustering algorithms (the subject we are interested in). Fourth column stresses the links with hard clustering techniques.

It may be noticed that when the partition is constrained to be hard in the optimization, yielding the so-called NCEM algorithm (Ambroise 1996), one obtains exactly the variant of ICM proposed by Besag that re-estimates the parameters of the classes during the clustering process (Besag, 1986). This variant was however not given a formal justification in probabilistic terms. This note shows that this variant of ICM is related to an EM algorithm applied to a hidden MRF model, much in the same way as the means algorithm is related to an EM algorithm applied to a mixture of Gaussian distributions (by hardening the classification when optimizing).

After these concepts of spatial clustering, and emphasizing on the MRF model, the remainder of this work is organized as follows. In Chapter 4 the development of the proposed method is provided. Chapter 5 presents computer simulations on the synthetic test images . Finally, conclusions and future work are given in in Chapter 7.

Chapter 4

Spatial Fuzzy Clustering with Simultaneous Estimation of Markov Random Field Parameters and Class

4.1 Introduction

As we saw in the previous Chapter, one of the widely used methods for image segmentation is based on Markov random fields (MRFs). MRFs model the objects in the image using a probabilistic model. Then, the segmentation task consists of classifying the pixels as well as estimating the parameters of the probabilistic model for each object. Segmentation results can be further improved when prior information is used in a Bayesian framework. This can be accomplished by using a penalty term that transforms the problem into a maximum *a posteriori* (MAP) problem.

The classical hard MRFs are successfully used in many areas, such as image restoration, image segmentation (as in our case), and volumetric object reconstruction. However, in several real-world image segmentation problems, the hard classification is not the most appropriate approach for such problems. Recently, fuzzy logic and statistics describing the uncertainty of segmentation and classification are used for segmentation tasks.

Generally speaking, methods using non-fuzzy MRF and fuzzy MRF available to this date consist of two main steps. The first is to estimate the model parameters used in the Markovian distribution of the random field. The second step is to perform the segmentation based on the parameters obtained in the first step. In this work, we present a strategy based on FMRF model: the Markovian distribution parameters are defined as functions of the degree of class membership, then the new cost function depends only on the class parameters, and can numerically be found using a nonlinear conjugate gradient method.

This Chapter is organized as follows: after this introduction, we describe the basics of FMRF, and the proposed method is presented in the following section.

4.2 Fuzzy MRF model

Fuzzy segmentation of an image is based on allowing each pixel to belong simultaneously to more than one class. Thus, the fuzzy segmentation problem is to associate to each pixel at location i a vector

$$(\alpha_{i1}\ \alpha_{i2}\ \dots\ \alpha_{iK}) \in [0,1]$$

with

$$\alpha_1 + \dots + \alpha_{iK} = 1 \qquad \forall i \tag{4.1}$$

where α_{ik} represents the degree of membership of a pixel to class k, and K is the number of classes. When $\alpha_{ik} = 1$, the pixel purely belongs to class k. Each pixel is represented by a vector in contrast to hard segmentation, in which each pixel is represented by a scalar indicating its class. Therefore, we obtain a matrix representing the grade of membership of a pixel for each class $\boldsymbol{\alpha}$

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1K} \\ \alpha_{21} & \dots & \dots & \alpha_{2K} \\ \dots & \dots & \dots & \dots \\ \alpha_{N1} & \dots & \dots & \alpha_{NK} \end{bmatrix}$$

In the MRF model, we consider two random fields, one for $\boldsymbol{\gamma}$, the segmentation results we want to obtain, and ant other one for the measured data $\boldsymbol{Y}$. The segmentation result we want to obtain is the realization $\boldsymbol{\gamma} = \boldsymbol{\alpha}$ of the field $\boldsymbol{\gamma}$. The measured data , i.e., the set of multispectral images, is a realization $\boldsymbol{Y} = \boldsymbol{y}$ of $\boldsymbol{Y}$, $\boldsymbol{y}_i$ is the feature vector of the pixel at location i. The joint distribution of the data $\boldsymbol{Y}$ and segmentation $\boldsymbol{\gamma}$ is:

$$P_{\boldsymbol{\gamma},\boldsymbol{Y}}(\boldsymbol{\alpha}, \boldsymbol{y}) = P_{\boldsymbol{\gamma}}(\boldsymbol{\alpha}) P_{\boldsymbol{Y}|\boldsymbol{\gamma}}(\boldsymbol{y}|\gamma) \tag{4.2}$$

where $P_{\boldsymbol{\gamma}}(\boldsymbol{\alpha})$ is the prior distribution of $\boldsymbol{\gamma}$ assumed to be stationary and Markovian, and $P_{\boldsymbol{Y}|\boldsymbol{\gamma}}(\boldsymbol{y}|\gamma)$ is the posterior distribution.

The problem of classifying the measured data $\boldsymbol{y}$ is to estimate the class parameters for each pixel, which can be accomplished by estimating the hidden field $\boldsymbol{\gamma}$ from the observed field $\boldsymbol{Y}$. In a Bayesian framework, suppose $C(\hat{\boldsymbol{\alpha}}, \boldsymbol{\alpha})$ is the cost of choosing an estimate $\hat{\boldsymbol{\alpha}}$ when $\boldsymbol{\alpha}$ is the truth. The estimator $\hat{\boldsymbol{\alpha}}$ of $\boldsymbol{\alpha}$ is obtained by minimizing this cost function. Then segmentation consists of (i) defining a cost function $C(\hat{\boldsymbol{\alpha}}, \boldsymbol{\alpha})$, (ii) finding the optimum estimator $\hat{\boldsymbol{\alpha}}$ of $\boldsymbol{\alpha}$ by minimizing the cost function. If we select our cost function as the negative of the likelihood plus a penalty term, we obtain the MAP estimation given by:

$$\hat{\boldsymbol{\alpha}} = \arg\max_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha}|y) = \arg\max_{\boldsymbol{\alpha}} \frac{p(\boldsymbol{y}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})}{p(\boldsymbol{y})} \tag{4.3}$$

Since the prior probability $p(\boldsymbol{y})$ is independent of $p(\gamma)$, then the maximization of Eq. 4.3 is equivalent to the maximization of $p(\boldsymbol{y}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})$. Then we have:

$$\begin{aligned}\hat{\gamma} &= \arg\max_{\boldsymbol{\alpha}} \left[p(\boldsymbol{y}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})\right] \\ &= \arg\max_{\boldsymbol{\alpha}} \left[\log p(\boldsymbol{y}|\boldsymbol{\alpha}) + \log p(\boldsymbol{\alpha})\right]\end{aligned} \tag{4.4}$$

where the cost function is $C(\hat{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) = -\log p(\boldsymbol{y}|\boldsymbol{\alpha}) - \log p(\boldsymbol{\alpha})$.

Depending on the choice of $P_{\gamma,\boldsymbol{Y}}(\boldsymbol{\alpha}, \boldsymbol{y})$, different MRFs and segmentation methods can be constructed. A common probabilistic model used is the Gaussian distribution, but extension to other distributions is straightforward by simply using the corresponding PDFs. Choosing a Gaussian distribution the MAP estimate can be obtained by maximizing:

$$\begin{aligned}L(\boldsymbol{\alpha}) = \sum_{i=1}^{N}\Bigg\{\log\Bigg[\alpha_{i1}\frac{1}{(2\pi)^{\frac{M}{2}}|\boldsymbol{\Sigma}_1^2|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(\boldsymbol{y}_i-\boldsymbol{\mu}_1)^T(\boldsymbol{\Sigma}_1^2)^{-1}(\boldsymbol{y}_i-\boldsymbol{\mu}_1)\right)+\cdots \\ +\alpha_{iK}\frac{1}{(2\pi)^{\frac{M}{2}}|\boldsymbol{\Sigma}_K^2|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(\boldsymbol{y}_i-\boldsymbol{\mu}_K)^T(\boldsymbol{\Sigma}_K^2)^{-1}(\boldsymbol{y}_i-\boldsymbol{\mu}_K)\right)\Bigg]\Bigg\}+\log p(\boldsymbol{\alpha})\end{aligned} \tag{4.5}$$

For a fuzzy MRF model α_{ik} denote the degree of membership of pixel i to class k, and $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k^2$ are the MRF parameters that define all distributions of $\boldsymbol{y}$ conditional on $\boldsymbol{\alpha}$.
To construct our final cost function, we need to choose a penalty term. A Gibbs penalty term that is commonly used in based on the prior knowledge that pixels tend to belong to the same class as their neighbors:

$$p(\boldsymbol{\alpha}) = \frac{1}{Z}\exp\Big(-U(\boldsymbol{\alpha})\Big) \tag{4.6}$$

with

$$U(\boldsymbol{\alpha}) = \beta\sum_{i=1}^{N}\sum_{j\in C(i)} V_C(\boldsymbol{\alpha}) \tag{4.7}$$

where $U(\boldsymbol{\alpha})$ represents the energy function, V_C is a function of $\boldsymbol{\alpha}$ defined on the clique C, and the normalizing constant Z is called the partition function. A clique is defined as a set of pixels, which consists of either a single pixel or a group of pixels. In this work, we use a 4×4 neighborhood around the particular pixel as its clique, because after some experiments we realize that it is a good trade between accuracy and computing cost.

4.3 Proposed algorithm

The main idea of the proposed method is to segment images by Fuzzy Markov random field method. Our key contribution is that, instead of estimating the means and the variances as a pre-processing step, we develop a method where there is no need to explicitly estimate them, because they are implicitly estimated simultaneously while performing classification. In this way, the estimation of the classes is not steered by the erroneous mean and variance estimates.

To accomplish this, expressions that are functions of degree of class membership $\boldsymbol{\alpha}_k$ replace the means and variances

$$\hat{\mu}_{mk} = \frac{1}{\sum_{i=1}^{N} \alpha_{ik}} \sum_{i=1}^{N} \alpha_{ik} y_{im} \qquad \hat{\sigma}^2_{mk} = \frac{1}{\sum_{i=1}^{N} \alpha_{ik}} \sum_{i=1}^{N} \alpha_{ik} (y_{im} - \mu_{mk})^2 \tag{4.8}$$

$$\boldsymbol{\mu}_k = \begin{bmatrix} \mu_{1k} \\ \mu_{2k} \\ \dots \\ \mu_{Mk} \end{bmatrix} \qquad \boldsymbol{\Sigma^2_k} = \begin{bmatrix} \sigma^2_{1k} & 0 & \dots & 0 \\ 0 & \sigma^2_{2k} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \sigma^2_{Mk} \end{bmatrix} \tag{4.9}$$

where $k = 1, ..., K, K$ is the number of classes, N the total number of pixels in the image, $\boldsymbol{y}_i$ the feature vector of the ith pixel, y_{im} its mth element, and α_{ik} is the ith component of $\boldsymbol{\alpha}_k$. When we closely look at these equations, we can see that $\boldsymbol{\mu}_k$ values are mostly affected by pixels from class k since α_{ik} will have values close to 1 for these pixels and α_{ij}, $\forall j \neq k$, will have values close to 0. These equations then calculate means and variances for each class based on all pixels but weighting each pixel with respect to its membership to a certain class. The advantage of our method is the ability of write an expression for the means and variance for each class based on fuzzy membership of all pixels, then, the only variables that affect the cost function are the α_{ik}s.

The energy term corresponding to the *a priori* model described in the previous section is: $p(\boldsymbol{\alpha}) = \frac{1}{Z} \exp -U(\boldsymbol{\alpha})$, and in our case, selecting the Gaussian MRF model which is widely used in many applications for the energy function:

$$U(\boldsymbol{\alpha}) = \beta \sum_{i=1}^{N} \sum_{k=i}^{K-1} \sum_{j \in V_i} \left(|\alpha_{ik} - \alpha_{jk}|^2 \right) \tag{4.10}$$

where V_i represents the pixels belong to the i's neighborhood.

Since $\alpha_{1i} + \alpha_{2i} + \ldots + \alpha_{Ki} = 1$, thus $\alpha_{Ki} = 1 - \sum_{k=1}^{K-1} \alpha_{ki}$, and substituting Eqs. 4.6 and 4.10 into 4.5, we can obtain the segmentation result $\boldsymbol{\alpha}$ as:

$$\begin{aligned} \hat{\boldsymbol{\alpha}} &= (\hat{\boldsymbol{\alpha}}_1 \ \dots \ \hat{\boldsymbol{\alpha}}_{K-1}) \\ &= \arg\max_{\hat{\boldsymbol{\alpha}}} \sum_{i=1}^{N} \Bigg\{ \log \Bigg[\alpha_{i1} \frac{1}{(2\pi)^{\frac{M}{2}} |\boldsymbol{\Sigma}^2_1|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (\boldsymbol{y}_i - \boldsymbol{\mu}_1)^T (\boldsymbol{\Sigma}^2_1)^{-1} (\boldsymbol{y}_i - \boldsymbol{\mu}_1) \right) + \cdots \\ &+ \alpha_{i(K-1)} \frac{1}{(2\pi)^{\frac{M}{2}} |\boldsymbol{\Sigma}^2_{(K-1)}|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (\boldsymbol{y}_i - \boldsymbol{\mu}_{(K-1)})^T (\boldsymbol{\Sigma}^2_{(K-1)})^{-1} (\boldsymbol{y}_i - \boldsymbol{\mu}_{(K-1)}) \right) \\ &+ \left(1 - \sum_{k=1}^{K-1} \alpha_{ik} \right) \frac{1}{(2\pi)^{\frac{M}{2}} |\boldsymbol{\Sigma}^2_K|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (\boldsymbol{y}_i - \boldsymbol{\mu}_K)^T (\boldsymbol{\Sigma}^2_K)^{-1} (\boldsymbol{y}_i - \boldsymbol{\mu}_K) \right) \Bigg] \\ &+ \beta \sum_{k=1}^{K-1} \sum_{j \in V_i} |\alpha_{ik} - \alpha_{jk}|^2 \Bigg\} \end{aligned} \tag{4.11}$$

where $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k^2$ are functions of $\boldsymbol{\alpha}_k$.
The steps of the proposed algorithm can be summarized as:

1. Initialize the class parameters, that is $\boldsymbol{\alpha}(0)$.
2. To obtain the update $\hat{\boldsymbol{\alpha}}(n+1)$, the segmentation result of the current iteration $\hat{\boldsymbol{\alpha}}(n)$ is updated by maximizing Eq. 4.11 using the conjugate gradient method.
3. Repeat step 2 until a stopping criteria is satisfied.

In step 2, the numerical optimization is performed for the nonlinear Conjugate Gradient Method (CGM)[10][50]. The initial search direction $\boldsymbol{D}(0) = (\boldsymbol{d}_1(0)\ \dots\ \boldsymbol{d}_k(0))$ for CGM is set to be the gradient direction for each class

$$\begin{aligned} \boldsymbol{g}_k &= (g_{1k}, g_{2k}, \dots, g_{Nk})^T \qquad\qquad g_{ik} = \frac{\partial C(\boldsymbol{\alpha}_k)}{\partial \alpha_{ik}} \\ \boldsymbol{G} &= (\boldsymbol{g}_1\ \boldsymbol{g}_2\ \dots\ \boldsymbol{g}_K) \end{aligned} \tag{4.12}$$

The search direction for class k for the nth iteration $\boldsymbol{d}_k(n)$ is

$$\boldsymbol{d}_k(n) = \boldsymbol{g}_k(n) + \eta_k(n-1)\boldsymbol{d}_k(n-1) \tag{4.13}$$

$$\eta_k(n-1) = \frac{(\boldsymbol{g}_k(n) - \boldsymbol{g}_k(n-1))^T \boldsymbol{d}_k(n)}{\boldsymbol{g}_k(n-1)^T \boldsymbol{d}_k(n-1)}$$

$$\boldsymbol{D}(n) = (\boldsymbol{d}_1(n)\ \boldsymbol{d}_2(n)\ \dots\ \boldsymbol{d}_K(n))$$

where $\boldsymbol{G}(n)$ is the gradient map at nth iteration, calculated by Eq. 4.12. Then, we have

$$\boldsymbol{\alpha}_k(n+1) = \boldsymbol{\alpha}_k(n+1) + \delta(n)\boldsymbol{d}_k(n) \tag{4.14}$$

The step size δ is determined by a line search method.

Once the fuzzy parameters are obtained, we determine hard classes by assigning each i pixel to the class k which has a higher α_{ik}.

We should note that the CGM does not guarantee a global convergence and have a few parameters that should be selected *a priori*. However, in our experiment, and simulations, we observed that starting from a random class labels produces more reasonable estimates than if we start from a uniform class labels. The only parameters that is required to be selected are the initial step size δ, and the smoothing parameter β. Using a small initial step size is a common practice, and the optimum step size is approximated by doubling the step size until the cost function increases, at which point the step size is fixed, and the update is performed. The parameter β affects the smoothness of the results, and serves as a parameter which the user can adjust to produce results that tend to eliminate segments that are smaller than a certain size. The estimation of the parameter β is difficult, which is addressed in detail in[34]. Some classic parameter estimation methods include: coding method, maximum likelihood[6], pseudo likelihood[7][8], least squares fit[17][45] and mean field approximations[59][18][60]. However, in this work we opt to determine β

heuristically based on a number of prior simulations. The obvious drawbacks in the determination of this parameter include the computation time taken, the fact that β has to be deduced for each different data set, and the fact that admittedly the choice of β may be subjective. On the other hand, it is easy and does not need assumptions.

The stopping criterion we use for CGM is based on the norm of the gradient image, and we stop iterations when this norm is below a certain threshold.

Chapter 5

Experimental Work

5.1 Simulated Multispectral Image

To illustrate the proposed method, we first applied our method to a computer simulated multispectral image as shown in Fig. 5.1. The size of the image is $60 \times\ 60 \times 3$; for each pixel, the feature vector contains three elements, which contains a different value according to the distribution of the class it belongs to. These values correspond to different color intensities when the image is plotted.

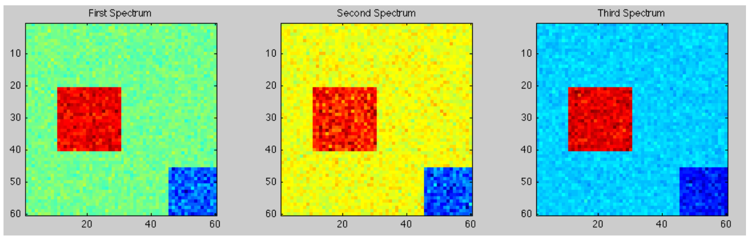

Figure 5.1: simulated Multispectral Image

As it is easy to see, the simulated image consists of three regions: three different multidimensional Gaussian distributions, with the following mean vectors and covariance matrices[1]:

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 10 \\ 2 \\ 20 \end{bmatrix} \qquad \boldsymbol{\mu}_2 = \begin{bmatrix} -5 \\ -6 \\ -10 \end{bmatrix} \qquad \boldsymbol{\mu}_3 = \begin{bmatrix} -15 \\ -20 \\ -20 \end{bmatrix}$$

$$\boldsymbol{\Sigma}_1^2 = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix} \qquad \boldsymbol{\Sigma}_2^2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{\Sigma}_3^2 = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

[1]Note that these parameters are unknown when we perform the segmentation.

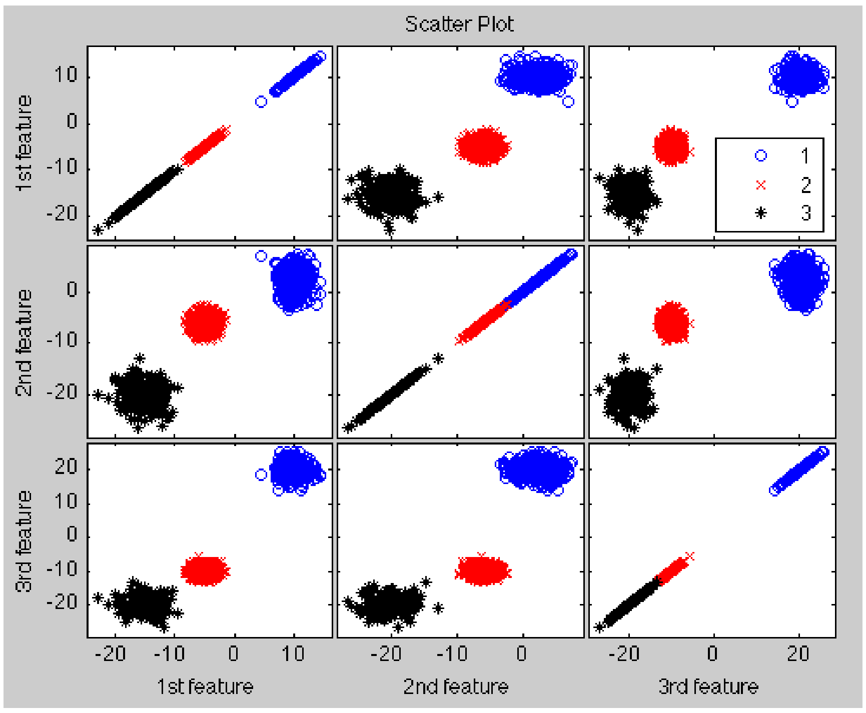

Figure 5.2: Scatter Plot

If we plot the data in a scatter plot[2], Fig. 5.2, comparing each feature against the others, the three clusters are easily distinguishable. Neither of the features seem to be correlated between them (of course, apart from one vs. itself). But it shows the nonlinear relationships between the observations. Furthermore, since the data is represented by a mixture model of simple relationships, these relationships are visually evident as superimposed patterns.

In this case, the number of classes, three, are supposed known. The initial $\boldsymbol{\alpha}$ matrix is randomly found, always satisfying the restrictions in Eq. 4.1. After some experiments, trying different values for the parameter β, we realized that the results are similar although this parameter is changed, eventually it is set to be 0.3 as a suitable value. We obtain results as shown in Fig. 5.3 after CGM iterations are complete. To do a performance comparison, the classical fuzzy C-means, K-medoids (as a K-means variation), and the Mixture of Gaussians (all three explained in subsection 2.1.3, page 30) methods are also applied to the same image.

[2]See subsection 1.3.2, page 15.

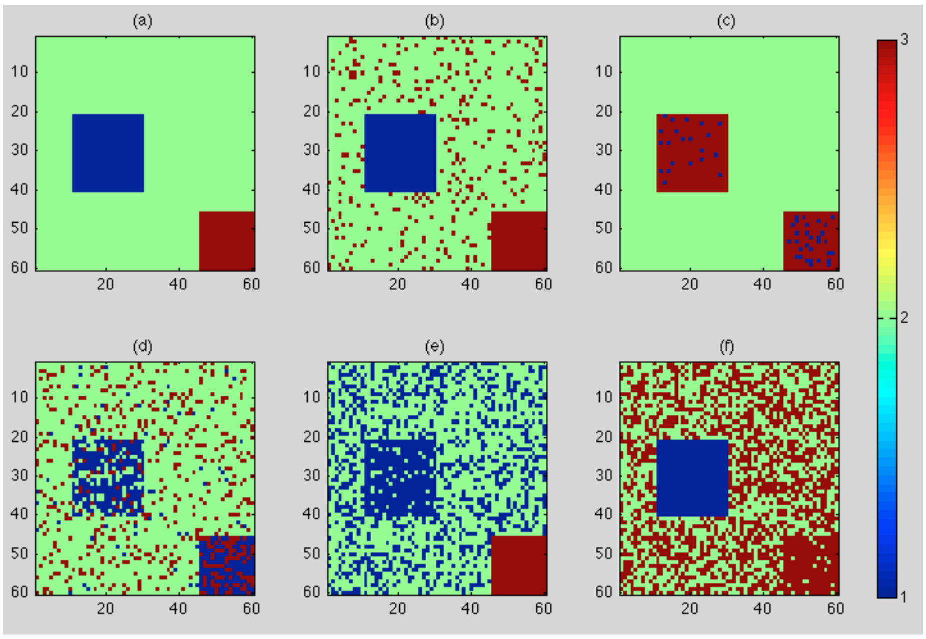

Figure 5.3: Segmentation of the test image. (a) True segmentation, (b) proposed method with random initial membership, (c) proposed method with uniform initial membership, (d) Mixture of Gaussians, (e) K-medoids, and (f) Fuzzy C-means

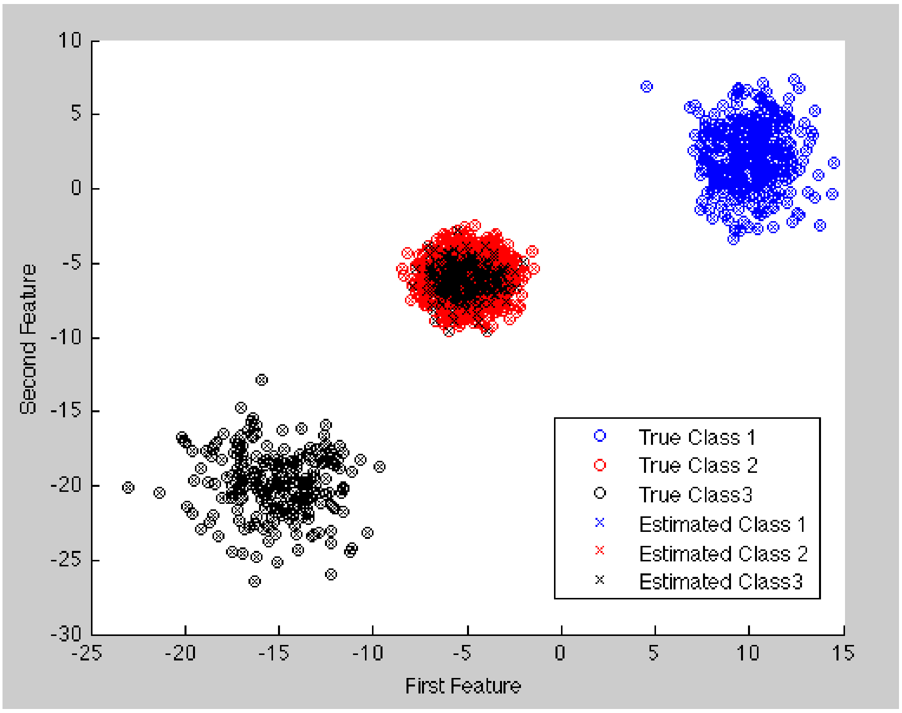

Figure 5.4: Estimated vs. True Clusters

Looking at Fig. 5.3, we can see that improved results are obtained with our method rather than with the classical ones. If we compare (b) and (c), we realize that choosing an uniform initial $\boldsymbol{\alpha}$ (the initial membership of each pixel), the method hardly distinguish between class one and three, consequently we conclude that it is more reasonable to work with a random initialization. We infer that somehow, this uniform initialization lead the algorithm to not be able to clearly differentiate between class one and three. Finding out the reason why the algorithm behaves like that could be an interesting objective in a future work.

On the other hand, if we plot the data, assigning the first feature to the horizontal axis vs. the second feature in the vertical axis, and representing with dots the true clusters and with crosses the estimated with method (b), we observe that there are some misclassified pixels, that belong to class two but they are classified as class three, see Fig. 5.4; we also can observe it in Fig. 5.3-(b).

In addition to visual evaluation we perform a quantitative analysis of the results: the *confusion matrix* and the *dice coefficient*, as defined in subsection 1.3.3, page 16.

To assess the method's reproducibility, and its statistical properties, we apply to the same multispectral image 100 times the proposed method, Table 5.1 shows the confusion matrix of the estimated labels showed in Fig. 5.3-(b) and Table 5.2 the mean of the different confusion matrices obtained after the 100 experiments. Notice that Table 5.1, as well as Fig. 5.3-(b) and Fig. 5.4, represent the confusion matrix of an average segmentation result of the proposed method; i.e. a clustering result that has an accuracy of 0.9 (see Table 5.3, where the mean of the dice coefficient, equivalent to the accuracy, is shown). In some experiments we get a perfectly labeled image (accuracy=1).

		True		
		ω_1	ω_2	ω_3
Estimated	ω_1	400	0	0
	ω_2	0	2648	0
	ω_3	0	327	225

Table 5.1: Confusion Matrix Estimated Image

		True		
		ω_1	ω_2	ω_3
Estimated	ω_1	327	42	43
	ω_2	12	2757	8
	ω_3	61	176	174

Table 5.2: Confusion Matrix of the proposed method

Looking at Table 5.1 , we appreciate the same as in Figs. 5.3-(b) and 5.4, i.e. the labeling result is nearly perfect, except from the misclassified pixels from class two which are estimated to belong to class three. From a wider point of view, Table 5.2, representing the average of 100 segmentations, shows that, though there are some misclassifications in each class (the value of these misclassifications is very low), most of the misclassifications happen between class two and three (pixels belonging to class two are classified as class three), as we have already observed visually.

The explanation of the higher misclassification value between classes two and three can be extracted from the feature information. If we observe Fig. 5.2, specifically the scatters that relate feature three against the other two features, we realize that cluster two and three are quite close to each other. In fact, displaying feature three vs. itself, the two clusters are overlapped. That means that the values of the pixels in this feature are similar; i.e., if feature three of the multispectral image represents an image characteristic, either can be intensity, or, as we will see in the next section, if the image represents a map, it can be the type of street, this characteristic could be the same or very similar for these pixels. This makes the differentiation between these two classes more difficult.

In addition, Fig. 5.1 corroborates this assumption, inasmuch as the pixels belonging to class two and three have a similar color (from dark to light blue), which means that they have a similar value (intensity value). Besides, since we know the means and variances of the Gaussian distributions used to simulate the data, see that although the third feature of $\boldsymbol{\mu}_2$ and $\boldsymbol{\mu}_3$ are not remarkably close, the value of $\boldsymbol{\Sigma}_3$ is quite high, what makes pixels belonging to this class to have spreader values and get close to the values of the ones belonging to the second class.

After 100 experiments, also applying fuzzy C-means method, K-medoids method, and Mixture of Gaussians method, the mean and the standard deviation of the Dice measure are also calculated, the results are shown in Table 5.3. In this case, the dice coefficient is equivalent to the accuracy (true labeled pixels over all the pixels in the data set), therefore, it represents a reliable measure of evaluation.

Method	Mean	Std. Devi.
Proposed Method Random Initialization	0.9042	0.1099
Proposed Method Uniform Initialization	0.8864	0
Mixture of Gaussians	0.7928	0
K-medoids	0.7038	0.1482
Fuzzy C-means	0.6810	0.0638

Table 5.3: Comparison of the mean and the standard deviation of DC

These quantitative results also show that the proposed algorithm yields an improved performance. The proposed method, even using an uniform initialization performs better than other standard clustering methods. Though the standard deviation[3] in fuzzy C-means has a lower value than the other methods, which means that the different segmentation results (over the 100 experiments) are more similar between them than the ones obtained from another method, the dice coefficient (consequently, the accuracy) is significantly lower.

On the other hand, these results are consistent with the visual results we get in Fig. 5.3. In the same way as with our method, in segmentations (d), (e), (f) (which also represent average results, the ones that have the accuracy shown to be the mean between all the segmentation results for each of the different methods) the most part of the misclassifications occur between class two and three. With the standard methods, we get blurrier results, like noisy images. The better performance of the proposed method is due to the inclusion of the spatial information, that constraints two neighboring observations more likely to belong to the same group than two observations lying far apart. Therefore, our method gets more accurate results.

Another matter to emphasize is the fact that the proposed method does not require a high number of iterations (in one experiment), consequently, it is not time consuming. Actually, it needs less iterations than some of the standard methods.

Finally, we have to add the fact that the algorithm is significantly sensitive to the initial randomly selected $\boldsymbol{\alpha}$ matrix (like some of the clustering methods, e.g. K-means); we run multiple times to reduce this effect.

Notice that in these experiments we know the true labeling. Thus, we are able to evaluate in a quantitative way the performance of the proposed method, which has proved to be strongly reliable. However, this information is not used for the clustering process, since we are working with unsupervised methods.

In the following section we test our method with data which we do not have the labels. In this way, we apply the algorithm to a more realistic situation. In addition, we cannot validate the results with the same evaluation measures used in this section; consequently, the evaluation is made through some clustering validation measures (see subsection 2.2.3, page 42).

5.2 Simulated Streets Data

Operationally, numerous street data are interpreted daily in support of crime prediction/prevention. One important application of these crime data, is to classify (segment) the different regions depending on their characteristics/patterns, and understand how they affect to the probability of a crime event. In other words, having a map image, it is very interesting to develop an algorithm capable of finding regions with similar characteristics; being these data the type

[3]In the proposed method with uniform initialization the standard deviation is 0, since the initialization in each iteration is always the same. Therefore, the segmentation results are the same.
In the Mixture of Gaussian method the results are also always the same due to the development of the code.

of street, business locations, public transport stations, hospital locations, crime events (burglaries, homicides, rapes, etc.), for example; these patterns help to find some kind of links between regions[4]. Afterwards, it will be easier to find a suitable prediction algorithm that fits to each region, and, therefore, the prediction (in this case, the crime prediction, i.e. the probability of a crime event) will be more accurate.

In the future, computer assisted segmentation and prediction techniques are expected to contribute to this task.

Anyway, in this work we only are concerned about the first step, dividing a map image into different regions, clusters in our terminology. The more advanced predicting task, that follows the segmentation step, at this point, is being developed for other people from our group.

In order to prove the validity of the proposed method to this map segmentation purpose, there are two more types of images considered for segmentation. One kind, with which we work in this section, is also 'simulated'. The proposed segmentation approach is applied to segment this simulated map image when only using the type of streets and the business density as image features. More features, like crime density, CTA locations, hospital locations, etc. should be used for more complicated mapping images. Thus, another type of image consists of an original map of the city of Chicago and these features location on it. More on this in the next Chapter.

Before the results, explain how the 'simulated' data, Fig. 5.5, is created.

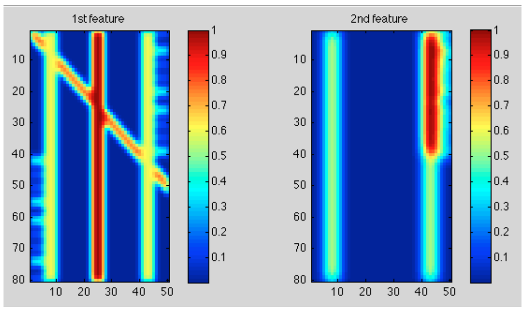

Figure 5.5: simulated Streets Image

The first feature of the multispectral image consists of the type of the street of the corresponding pixel location in the map image. From a pixel intensity value of 0 to 1 the classification is the following:

0	background (buildings, no streets in those locations).
0.2	little street.
0.4	avenue.
0.6	drive way.
0.8	express way.
1	highway.

[4]Which do not necessarily be side by side in the image. Notice the importance of clustering methods that are able to find non-homogenous shaped clusters.

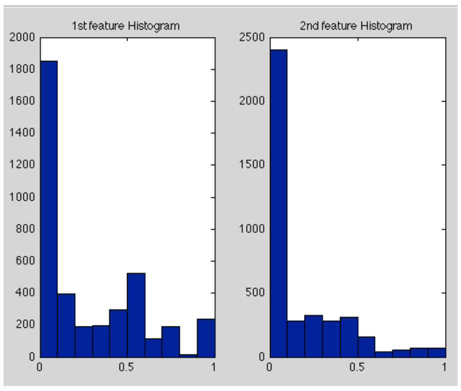

Figure 5.6: Streets Histogram

After the value assignment of each pixel, the image is smoothed. The reasons why we smooth the image are, on one hand, a smoothed image is more realistic than an image with sharper shapes, and on the other hand, due to algorithm performance.

The second image feature represents the business density, i.e. the number of stores, restaurants, company headquarters, and any kind of business, that are located in a specific region. From 0 to 1, the increasing intensity value represents a higher density of business activity. Of course, the image is also smoothed.

In that case, we want to find different clusters in this 'simulated' map that segment the image in regions where each region has pixels with similar type of streets and density of business.

Their histograms, pixel street density and pixel business density, are also shown in Fig. 5.6. Observing the left picture, which corresponds to the streets density, six modes can be clearly found, agreeing to the five different types of streets defined, besides the background (buildings) which can be considered as the sixth mode (the higher peak). The right picture of Fig. 5.6, which represents the business density of the different areas, three modes are differentiable in the histogram (non, low and high business activity).

In those circumstances, with all this visual information (Figs. 5.5 and 5.6), we have to infer the number of clusters that we expect to find, also trying to get a trade between the information given by the two features, as well as the information given by the pixels spatial location in the image.

Although it is a difficult and highly subjective task, the class number of the image is assumed to be five, The 'ground truth' is given in Fig. 5.7, corresponding to:

1. Background: locations which has neither streets, nor business areas.
2. Locations with medium-sized streets (as driveways) with business areas (but not highly busy areas).
3. Locations with main (bigger) streets (as highways or express ways) with business areas (but not highly busy areas).

4. Locations with little streets, and with business areas (but not highly busy ones).
5. Locations with medium-sized streets (as driveways) and highly busy areas.

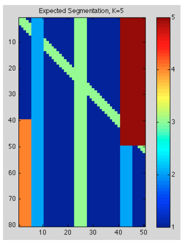

Figure 5.7: Expected Segmented Image for 5 classes

But as we said before, and it is explained in section 2.2, page 41, estimating the number of cluster is a hard accomplishment. Therefore, in this section, to infer the suitable number of clusters, we make several experiments splitting the data in a different number of clusters in each one. Apart from the five class ground truth we also simulate the classification expected for different values of K (number of clusters), Fig. 5.8:

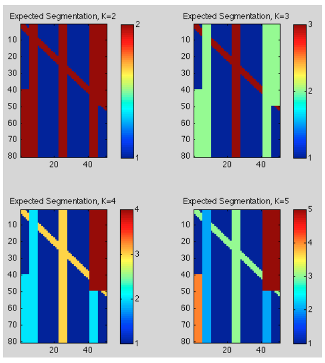

Figure 5.8: Expected Segmented Image for different *a priori* chosen number of clusters

K=2	– Background (buildings).
	– Any type of streets
K=3	– Background (buildings).
	– Little and medium-sized streets.
	– Big streets.
	Apart from the different size of the streets, we also expect this differentiation because in the third class is where the business are located.
K=4	– Background (buildings).
	– Little and medium-sized streets with lower business activity.
	– Big streets.
	– Little and medium-sized streets with higher business activity.

After that, besides the visual analysis, we also apply some clustering evaluation methods (see subsection 2.2.3, page 42), in order to have more information, and be able to find the most 'natural' number of clusters.

Apart from the proposed algorithm, all this procedure is also done applying the classical fuzzy C-means algorithms. Thus, we have a wider vision to estimate the number of clusters, and to check if the inferred number of clusters could be different depending on the applied method.

After the experiments, it is clearly seen that the proposed method segmentation result is much closer to the ground truth, and the result of our method is more homogeneous and smoother than the obtained with fuzzy C-means algorithm. The comparison of the proposed method with the standard fuzzy C-means is

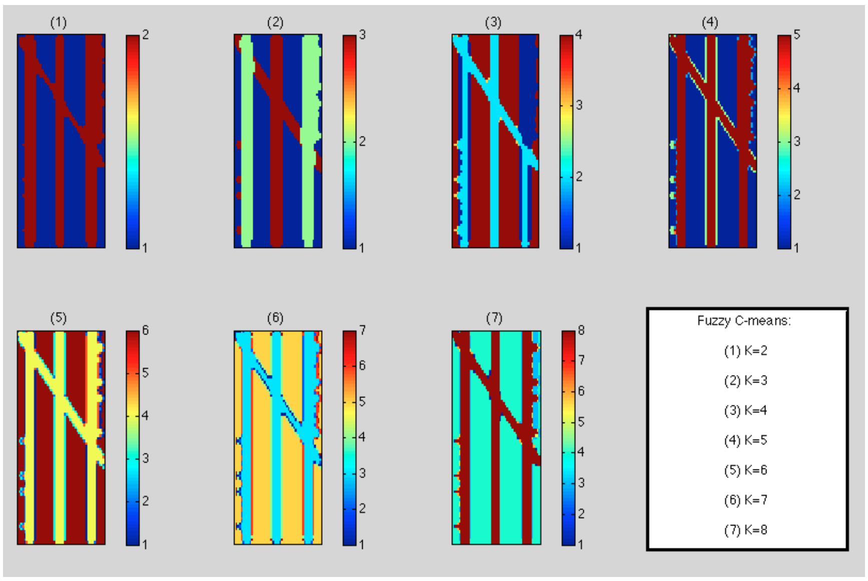

Figure 5.9: Segmented Image by fuzzy C-means method

shown in Figs. 5.10 and 5.9, respectively.

Fuzzy C-means clustering method does not take into account spatial relationship at all. Although, for $K = 2$, $K = 3$, and $K = 4$ the results are acceptable, Fig. 5.9 shows that the segmented images are not the ones expected. The regions are detected in (1) and (2) (background, bigger streets, smaller streets), but when the value of K is incremented the algorithm is not able to perform a suitable classification. Despite it detects the background and the streets as different classes, it randomly assigns pixels to the rest of classes, for no apparent reason, without taking into account the business location information, no matter the value of K.

In comparison, the proposed method gives reasonably good results.

As it is explained in Chapter 4, the determination of whether a pixel is classified as belonging to one cluster or to another is given by its joint fuzzy membership function as shown in Eq. 4.11, page 62. In this function, there are two components, one balancing the statistical and the other one the spatial

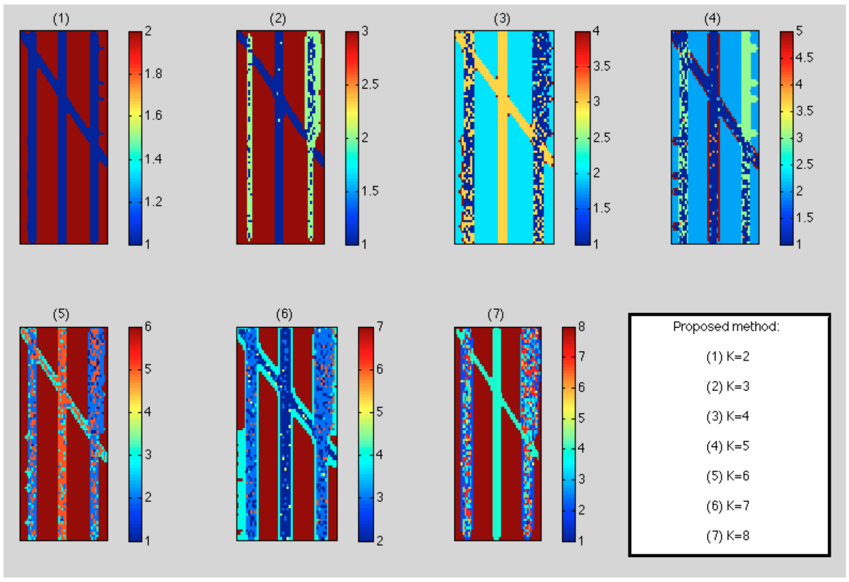

Figure 5.10: Segmented Image by the proposed method

information; the weight of each component is regularized by the parameter β, when β increases, the spatial contribution increases too. In this experiment, this parameter is decided heuristically based on a number of simulations in the study. By cross validating on a set of simulated data (previous section) with known labels, we found that setting β to be 0.3 is a good choice. In addition, instead of choosing β empirically, several methods can be adopted as discussed below in Chapter 7.

Fig. 5.10 shows the best results among these obtained by this method. As well as, it differentiates the bigger streets from the smaller ones, it successfully distinguishes the regions in the smaller streets where the business activity is higher, Fig. 5.10-(4). In addition in (5) and (6) it is able to discriminate the smallest streets (left-bottom and right top corners). Therefore, our method generate much more accurate results than the other algorithm.

However, the proposed method fails in the fact that as K increases, many small regions (holes) appear in the image. This is because the value of K is too high, thus it splits the most 'natural' clusters into these small areas, which give the appearance that the noise dominate these images. The differentiation between the small streets with high and low business activity is almost lost in the noise.

Observing Fig. 5.10-(6), we detect another shortcoming, the edges of the streets are classified as a cluster itself, also including the smallest streets. This is due to have smoothed the simulated image. From that, we suspect of smoothing the image, even though, at first seems to be a more realistic representation, is not suitable for the proposed method performance. Highlight that this effect is also detected when classifying with the standard fuzzy C-means method.

These experiments expose the difficulty of working with real data, when neither the labels, nor the number of cluster are known. *A priori*, we do not know how the segmentation will look like after the clustering procedure, we can only infer it. In fact, the aim of the unsupervised methods is to find the patter relations that we cannot because of the lack of information.

Consequently, it is very difficult to say if a segmentation is good or not. Therefore, to measure the segmentation validity, we apply the quantitative evaluation of performance by using the clustering evaluation methods explained in section 2.2.3, page 42. Specifically, the cluster cohesion, Eq. 2.37; cluster separation, Eq. 2.38; squared sum error, Eq. 2.39; and silhouette coefficient, Eq. 2.40, that are related to the gap statistic explained in subsection 2.2.2, page 42. Besides our method, two more methods are used for segmenting the image, the standard K-means and fuzzy C-means algorithms, in order to compare the three results through the clustering evaluation measures.

In Fig. 5.11 these parameters results, for the different segmentations (each

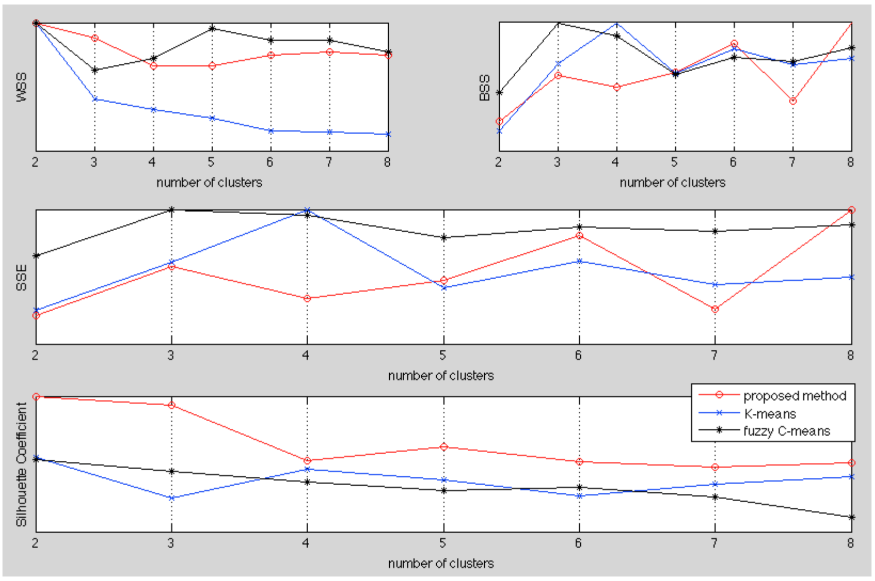

Figure 5.11: Clustering Evaluation

segmentation is done with a different number of clusters) applying the three different methods, are represented. It proves that the interpretation of these measures is not an easy task.

The K-means method has the lowest WSS, meaning the highest cluster compactness. K-means and fuzzy C-means have the highest BSS, which shows a higher distance between different clusters, but as K increases, the proposed method performance gets better. Although these parameters indicate that K-means and fuzzy C-means get more cohesive clusters and more separate ones from the others, Figs. 5.9 and 5.10 show that the proposed method results are closer to the expected segmentation.

On the other hand the silhouette coefficient support the performance of our method, since its higher value means a higher trade-off between the within and between cluster distance.

From this graphics we can also try to infer the optimal number of clusters, based on the gap statistic. For the K-means method it seems to be $K = 4$, since for this number of clusters there is a dip in the WSS representation and a peak in the other parameters graphics. In the case of fuzzy C-means, it could be $K = 3$ or $K = 5$, for $K = 3$ there are peaks for BSS and SSE, a knee for WSS and a dip for the silhouette coefficient, but for $K = 5$ there are also a peak for WSS and knees for the other parameters. Eventually, for the proposed method, the optimal value of K could be from $K = 4$ to $K = 6$, due to the peaks and knees observed for these values.
Because, in those circumstances, the 'natural' number of clusters is not evident, we have to use the visual information from the segmented images to reinforce the knowledge got from this parameters graphics. Finally, observing Figs. 5.9 and 5.10, we conclude that for fuzzy C-means $K = 3$ is a suitable value. However, for the proposed method the optimal one is $K = 5$, as we had thought it was supposed to be. Consequently, we deduce from these results, that the suitable number of clusters depends on the applied method.

Chapter 6

Chicago Police Department Project

6.1 Introduction

As we said, this work is only a little part of a more complex project, which is being developed in support of crime prediction/prevention. Right now, the *Illinois Institute of Technology* is working with the *Chicago Police Department* (CPD) in order to develop an automated, proactive predictive analytics system.

The aim of this bigger work is to plan a demonstration project that will apply advanced statistics and machine learning techniques to leverage a large set of data in order to identify connections across disparate datasets and discern subtle patterns of emerging violence that would not otherwise have been apparent; in other words, to develop a plan leading to the creation of a crime prediction model that will leverage information from all relevant sources to productively predict crime patterns and frequencies, with a special focus on the area of public violence.

This project offers a unique opportunity to demonstrate and evaluate the potential of applying predictive analytics to the problem of crime prevention in a large urban setting.

Within this project, an interesting analysis, as a previous step, is to perform an unsupervised (since we do not know which would be a suitable classification) segmentation of the city to find how many different types of places there are in terms of crime models. These should lead to obtain different areas not necessarily close to belong to the same cluster and thus to require the same model. One possible way to represent these clusters may be to color areas belonging to the same cluster with the same color; i.e. the development of graphical displays to assist in visualization of relationships among crimes and common patterns across neighborhoods.

Until now, we only have been working with simulated data; in order to get to a more realistic situation in this Chapter we work with real spatial data (geospatial data, specifically). Unfortunately, due to the difficulty of learning the software that enables us to work with that kind of data, some difficulties in getting the permissions to use the police data, and the lack of time, we could not test the proposed method to these real data.

However, we found very interesting to include some knowledge about working with geospatial data in this work. After understanding how this type of data works, the future research of the method presented in Chapter 4 could be focused on applying it to these police information sources. This information is represented, as we said before, by geospatial data.

Spatial data refers to data describing location, shape, and spatial relationships. Geospatial data is spatial data that is in some way georeferenced, or tied to specific locations on, under, or above the surface of a planet. The geospatial data can be voluminous, complex, and difficult to process. Mapping Toolbox functions (MATLAB) handle many of the details of loading and displaying data, and built-in data structures facilitate data storage. Therefore, in the following subsection a brief introduction about the Mapping Toolbox is presented. Nevertheless, the more you understand about your data and the capabilities of the toolbox, the more interesting applications you can pursue, and the more useful your results will be to you and others.

6.2 Mapping Toolbox

6.2.1 Mapping Toolbox Overview

Mapping Toolbox provides tools and utilities for analyzing geographic data and creating map displays. You can import vector and raster data from shapefile, GeoTIFF, SDTS DEM, and other file formats, as well as Web-based data from Web Map Service (WMS) servers. Since our investigation group is working with shapefiles, in this brief Mapping Toolbox introduction we only talk about that type of file format.

The toolbox lets you customize the imported data by subsetting, trimming, intersecting, adjusting spatial resolution, and applying other methods. Geographic data can be combined with base map layers from multiple sources in a single map display. With function-level access to all key features, you can automate frequent tasks in your geospatial workflow. Briefly summarized, the toolbox provides functionality in the following areas:

- Vector and raster data import and export from standard formats and specific data products.
- Data retrieval from Web Map Service (WMS) servers for customized geographic datasets and related metadata.
- Digital terrain and elevation model analysis functions, including profile, gradient, line-of-sight, and view shed calculations.
- Geometric geodesy, including distance and area calculations, 3D coordinate transformations, and more than 65 map projections.
- Utilities for converting units, adjusting spatial resolution, wrapping longitudes, and managing spatially referenced images and raster data.
- 2D and 3D map display, customization, and interaction.

6.2.2 What Is a Map?

Mapping Toolbox software manipulates electronic representations of geographic data. It lets you import, create, use, and present geographic data in a variety of forms and to a variety of ends. In the digital network era, it is easy to think of geospatial data as maps and maps as data, but you should take care to note the differences between these concepts.

In this toolbox, map data is any variable or set of variables representing a set of geographic locations, properties of a region, or features on a planets surface, regardless of how large or complex the data is, or how it is formatted. Such data can be rendered as maps in a variety of ways using the functions and user interfaces provided.

6.2.3 What Is Geospatial Data?

Geospatial data comes in many forms and formats, and its structure is more complicated than tabular or even nongeographic geometric data. It is, in fact, a subset of spatial data, which is simply data that indicates where things are within a given coordinate system. Such coordinate systems, however, are local and not explicitly tied or oriented to the Earths surface.

What sets geospatial data apart from other spatial data is that it is absolutely or relatively positioned on a planet, or georeferenced. That is, it has a terrestrial coordinate system that can be shared by other geospatial data. There are many ways to define a terrestrial coordinate system and also to transform it to any number of local coordinate systems, for example, to create a map projection.

Geodata is coded for computer storage and applications in two principal ways: vector and raster representations. It has been said that raster is faster but vector is corrector'. There is truth to this, but the situation is more complex.

Raster Geodata

Raster geodata maps data as a matrix (a 2-D MATLAB array) in which each row-and-column element corresponds to a rectangular patch of a specific geographic area, with implied topological connectivity to adjacent patches. Raster is actually a hardware term meaning a systematic scan of an image that encodes it into a regular grid of pixel values arrayed in rows and columns.

Vector Geodata

Vector data (in the computer graphics sense rather than the physics sense) take the form of sequences of latitude-longitude or projected coordinate pairs representing a point set, a linear map feature, or an areal map feature. Such data consists of lists of specific coordinate locations (which, if describing linear or areal features, are normally points of inflection where line direction changes), along with some indication of whether each is connected to the points adjacent to it in the list. In the Mapping Toolbox environment, vector data consists of sequentially ordered pairs of geographic (latitude, longitude) or projected (x,y) coordinate pairs (also called tuples). Successive pairs are assumed to be connected in sequence; breaks in connectivity must be delineated by the creation of separate vector variables or by inserting separators (usually NaNs) into the sets at each breakpoint. For vector map data, the connectivity (topological structure) of the data is often only a concern during display, but it also affects the computation of statistics such as length and area.

In the context of geodata, vector data means points, lines, and polygons that represent geographic objects. Vector geospatial data is used to represent point features, such as cities and landmarks; linear features, such as rivers and highways; and areal features, such as bodies of water and voting districts.

The data that we use are represented in this way.

6.2.4 Mapping Toolbox Geographic Data Structures

Mapping Toolbox software provides an easy means of displaying, extracting, and manipulating collections of vector map features organized in geographic data structures. A geographic data structure is a MATLAB structure array that has one element per geographic feature. Each feature is represented by coordinates and attributes. A geographic data structure that holds geographic coordinates (latitude and longitude) is called a geostruct, and one that holds map coordinates (projected x and y) is called a mapstruct. Geographic data structures hold only vector features and cannot be used to hold raster data.

Geographic data structures most frequently originate when vector geodata is imported from a shapefile. Shapefiles encode coordinates for points, multipoints, lines, or polygons, along with non-geometrical attributes. As it is explained before, these geometrical attributes refer to point features, such as cities and landmarks; linear features, such as rivers and highways; and areal features, such as bodies of water and voting districts. In the street density case, we work with lines', and its corresponding shapefile encodes their coordinates, as well as the type of street (non-geometrical attribute).

6.3 First Work with the Police Data

6.3.1 Chicago Streets Density Image

In order to work with de data provided by the CPD, our group have developed a program to compute the density of streets in a Chicago Map and generate a zoomed image of Chicago.

Loading the streets information (the location of the different types of streets in the city of Chicago and their location) from the corresponding shapefile, we convert geographic vectors (latitude, longitude) of the streets shapefile of the entire city to regular data grid and create the georeferenced vector. Then, we obtain a structure containing the vector data, its attribute and a field name for a unequivocal identification. With the information stored like this and some other modifications, the data is ready for being classified, in combination with other data sets with the same format, by the proposed method in Chapter 4. In other words, at the end of this transformation we get the data stored in a similar format as the one used in section 5.2.
Eventually, a 2D map is displayed.

Observing Fig. 6.1, we realize that the resulting image looks like the one simulated in section 5.2, of course, a more complex one.
All this procedure is done implementing a MATLAB code through the func-

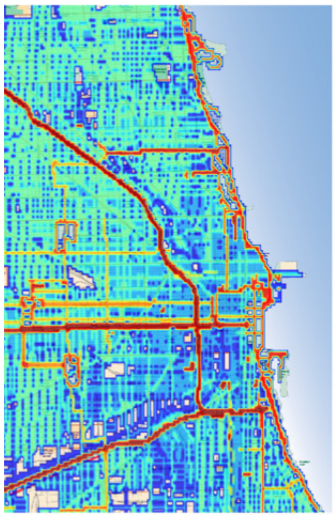
Figure 6.1: Chicago Streets Density Zoomed Image

tions provided by the Mapping Toolbox.
We do not list/explain them because it would be a long and hard task, and thoroughly explain the Mapping Toolbox is not a purpose of this work. Anyway, in the Mapping Toolbox User's Guide is where we find all the information about this tool needed to implement the algorithms used to work with this type of data sets.
(http://www.mathworks.com/help/toolbox/map/index.html)

6.3.2 Future Work

As well as the streets data, the CPD provides us other interesting data sets:

CTA location: the location of *Chicago Transport Authority* stations (public transport stations).

District boundaries: the boundaries of each district of the city of Chicago.

Gang boundaries: the boundaries that surround the areas where the different gangs are located.

Gang conflicts: where the gang crimes happened.

Parks: where the parks are located.

Pods: remote-controlled and viewable cameras, called Police Observation Devices, locations.

Schools: where the schools are located.

We should write a similar code as the one used to display and convert the streets density data, for the other data sets. After that, we will have the data prepared for the clustering procedure; i.e first as a multispectral image, and after the re-locations of the pixels as a matrix of feature vectors, each feature corresponding to different attributes (in this case, street type, is there is or there is not a park, etc.) of this pixel location.

As we said, we did not have enough time to test the proposed Spatial Fuzzy Clustering method using Markov Random Fields with this data sets, but future work will focus on that. Regarding the results of the experiments made in the previous Chapter, after applying the algorithm, we expect to get a useful city segmentation for being used in data pro-action, where predictive analytics are utilized to forecast possible future events. Since the overall goal is, through advanced algorithmic approaches, to create analyses, predictions, and data visualizations that could not readily be achieved by a human analyst.

Chapter 7

Discussion and Future Work

7.1 Conclusions

Throughout this Thesis we have been able to get an overview of the spatial unsupervised clustering problem and discussed the advantages and disadvantages of the current approaches.

We have used a top-down approach for the evaluation of the problem. This approach has been taken both for the study of the theoretical issues and for the design of the algorithm. For that reason, we have started considering spatial unsupervised clustering as a particular case of pattern recognition, then we have briefly emphasized the unsupervised learning, in Chapter 2, and we shortly reviewed the main unsupervised clustering methods.

Our contribution in this work is the application of a fully spatial fuzzy clustering scheme for segmenting multispectral spatial images. This is a generalized fuzzy clustering framework. With different similarity measurements and different derived expressions of the cluster representative parameter, the corresponding general Spatial Fuzzy Clustering (SFC) will behave differently[27]. Theoretically, any conventional fuzzy clustering method could be embedded into the framework, at least for obtaining an initial partition. SFC using Markov Random Fields (MRF) is just a special case which was implemented in this work. Therefore, about the different methods of spatial fuzzy clustering, in Chapter 3, we have mainly focused on the ones based on Markov Random Fields.

Chapter 4 presents a novel algorithm that is able to incorporate both local spatial contextual information and feature space information into the image segmentation, an unsupervised segmentation method for multispectral image segmentation based on Gaussian model and MRF model. Although many researchers have proposed the application of MRFs to model images in a Bayesian framework for image segmentation, to date, the MRF and fuzzy MRF parameters defining the Markovian distribution of the measured data are estimated separately, either by a preprocessing step, or by performing alternating optimization to estimate the model parameters together with the class parameters. Separating the estimation of these two groups of parameters would slow down the convergence, and would yield poorer results.

As we have fully explained, the proposed method considers the estimation of MRF parameters implicitly and combines the two tasks: estimation of MRF parameters and image segmentation together.

The images are assumed to be a MRF and we define a fuzzy energy neighborhood function to describe the interaction between neighboring pixels. The final labeling is determined by a joint fuzzy membership. Therefore, in deploying the proposed method, one has to assume that $P(\boldsymbol{Y}|\boldsymbol{\alpha})$ is Gaussian under $P(\boldsymbol{\alpha})$ being Markovian hypothesis, with $\boldsymbol{Y}$ the set of images and $\boldsymbol{\alpha}$ the labels (segmentation). Although, this assumption may not be always true in other situations, which may limit the applications of the method, the results of the experiments indicated that our method is superior to either Mixture of Gaussians, K-medoids, and the standard fuzzy C-means. The incorporated spatial information improved the performance of the clustering in reducing the 'blurry effect', and increasing the accuracy.

At this point, in Chapter 5, we have given some general impressions about the behavior of the our algorithm. A variety of images, including simulated and real images, were used to compare the performance of the algorithm. Experimental results show that the proposed method is effective and more robust than other clustering methods. In any case, in this Chapter we are not going to detail the results obtained as we have already done an evaluation in Chapter 5, where we have also explained and justified the results whenever it was possible.

Since there is usually no true labels available in practice, it is difficult to evaluate whether the proposed method is indeed the good or not in detecting the different regions. First, in our study, we evaluated our method using Gaussian simulated data, where we knew the 'ground truth', the true segmentation. Some other methods were taken into account in the method comparison: Mixture of Gaussians, K-medoids and fuzzy C-means. The first thing we have noticed is that the results are bolstered by both the visual and the quantitative results; that is to say, the plotted segmented image and the scatter plot as visual representations, and the confusion matrix and the dice coefficient as quantitative results. Regions are expected to occur at locations which are connected with the same Gaussian distribution, which exhibit a certain similar pattern; and also a single region is typically formed with neighboring pixels. For this image, concerning the traditional clustering algorithms tend to produce ambiguous membership, therefore, as comparable with the proposed, the classes of the neighbors of pixels should contribute more significantly to the final determination of the pixel's classification, which in turn requires a relatively larger β, the parameter which regularizes the spatial information weight in the criterion function.

A second remarkable fact is that this image segmentation, in particular, is easy to evaluate since the true labels are available to compare the segmentation results against the real ones. But practical applications are, however, increasingly required to work under unsupervised environment. That is to say, no training data is available and the validating procedure should be based on other unsupervised evaluation methods. Under such unsupervised environment, sometimes the models are not able to work consistently.

The same conclusion can also be drawn from the second type of image that has been tested, a multispectral image consisting of two features. The first feature represents the type of the street of the corresponding pixel location in the map image, the second image feature represents the business density that are located in a specific region.

When comparing the results against the ones obtained with fuzzy C-means, since we did not know the true number of clusters (K), we have found several segmentations with different values of K. After, we have plotted the segmented images, which have shown that the proposed method results are closer to the ones expected, the most 'natural' ones, which again indicates that our method is effective. Moreover, as K increased, our method has performed better than fuzzy C-means in controlling the way of splitting the clusters to new ones. Instead, fuzzy C-means has not been able to perform a suitable classification; despite it has detected the background and the streets as different classes, it randomly has assigned pixels to the rest of classes, for no apparent reason. Notice that, after these results, it could be inferred that the 'optimal' K is related to the method we are working with; in other words, depending on the clustering algorithm used for the segmentation procedure the choice of the suitable K could be a different one.

We accept that the proposed method has failed in the fact that as K increased, many small regions (holes) has appeared in the image. It has split the most 'natural' clusters into these small areas, which give the appearance that the noise dominate these images. The differentiation between the small streets with high and low business activity is almost lost in the noise.

The root cause could be that the MRF based segmentation model is very easily trapped in local optima due to the imposed spatial homogeneity constraint imposed by the region labeling component. As a result, the feature modeling component might not be able to learn the global parameters (i.e. $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ for each class). We should also note that the Conjugate Gradient Method (CGM), which compute the numerical optimization of the segmentation, does not guarantee a global convergence and have a few parameters, that should be selected *a priori*. This parameters are: δ, the step size; the initial $\boldsymbol{\alpha}$, which gives the first pixels membership assignment; and β, the regularization parameter.

The δ parameter is used when computing the CGM as a step size between iterations, i.e. it controls the weight of the membership change direction. It is successfully determined by a line search method. Using a small initial step size is a common practice, and the optimum step size is approximated by doubling the step size until the cost function increases, at which point the step size is fixed, and the update is performed.

We have become aware of some limitations due to the choice of the optimal $\boldsymbol{\alpha}$, since depending on this initialization very different segmentation results can be obtained. Although we have started with the premise of the randomly chosen produces more reasonable estimates than if we start from a uniform class labels, we have inferred that maybe, in some images, another initialization could be better. Which can lead us to think that there is no optima $\boldsymbol{\alpha}$ for all the situations, although we can find some compromising solution. For these reasons, we could think of a possible improvement implementing some kind of pre-processing training stage where the $\boldsymbol{\alpha}$ matrix should be estimated according to the testing images. Anyway, in our test images it has been proved that more suitable segmentations are obtained with a random $\boldsymbol{\alpha}$, rather than an uniform one.

Besides, another aspect to reconsider in our system is the choice of parameter β. In the method developed in this work, the determination of whether a pixel is classified as belonging to one cluster or to another is given by its joint fuzzy membership function as shown in Eq. 4.11, page 62. In this algorithm, the probability of a certain pixel belonging to a particular class is partially dependent on the probabilities of the class assignment of the neighbors. There are two components, one balancing the statistical and the other one the spatial information. The weighting parameter, β, as the power of the probability or the weight to its energy function, must be assigned to combine the two components in order to determine how much each component contributes to the whole system. With a constant weighting parameter, as in our case, segmentation results can fall into three cases. If the constant parameter makes the region labeling component dominant, the values of parameters estimated may deviate too much from the real feature data. If the constant parameter makes the feature modeling component dominant, the spatial relationship information is ignored in the final segmentation result. If a balance can be achieved between both components by choosing a proper constant parameter, the estimated parameters are not globally but locally optimal. All of these cases may generate inaccurate segmentation results. Some studies explores implementation schemes to combine the two components by introducing a variable weighting parameter[15].

In this work, this parameter is decided heuristically based on a number of simulations in the study. By cross validating on a set of simulated data with known labels, we found that setting β to be 0.3 is a good choice, but this cannot be taken as the optimal method. In addition, instead of choosing β empirically, several methods can be adopted, we discuss that in the following section.

One last remarkable thing in Chapter 5, is that we apply the quantitative evaluation of performance by using the unsupervised clustering evaluation methods explained in section 2.2.3, page 42. Specifically, the cluster cohesion, Eq. 2.37; cluster separation, Eq. 2.38; squared sum error, Eq. 2.38; and the silhouette coefficient, Eq. 2.40. Although useful, the information given by these measures is difficult to interpret. Emphasize that the silhouette coefficient support the performance of our method, since its higher value means a higher trade-off between the within and between cluster distance.

Finally, in Chapter **??** an automated, proactive predictive analytics system conducted by the *Illinois Institute of Technology* and the *Chicago Police Department* is presented, also is the role that our work could do as a previous step: perform an unsupervised segmentation of the city of Chicago, in order to find how many different types of places there are in terms of crime models; these should lead to obtain different areas not necessarily close to belong to the same cluster and thus to require the same predicting model. Unfortunately, we could not test our method to the data provided by CPD. However, we found very interesting to include some knowledge about working with geospatial data in this work. After giving a brief survey of this project, it is also described the software used for this purpose and the first experiments that have been conducted, the representation of geospatial data.

To sum up, the objectives of this study have been accomplished. The successfully achieved goals of this Thesis are:

· We have presented the work done using spatial fuzzy clustering based on Markov random fields for segmenting images.

· We have developed a method that estimates the parameters defining the Markovian distribution of the measured data while performing the data clustering simultaneously.

· We have used two different types of image data in order to test it.

· With both simulated multispectral image we have got good results, as the visual results, as well as, the quantitative evaluation measures boost.

· We have presented a bigger project in which our work could play a role, as a example of the use of the presented method.

Of course, more work can be done in this field, and more data has to be used to develop a complete system.

7.2 Future Work

This work makes an initial assertive step towards the study of spatial fuzzy clustering using Markov random fields. During the execution of the Thesis, some interesting ideas have been set out in order to continue with the research.

The experiments carried out in the Thesis have been only a preliminary approach to the problem. More experimentation is needed in order to come to solid conclusions. Although the experimental results on simulated test images show the effectiveness of the proposed method, it is necessary to work with other databases and asses the performance of the system in other situations.

Future perspective of works include evaluating more rigorously the heuristic to estimate the spatial coefficient β an understand its mechanism, formulating and testing the approach for different prior an conditional distributions, and validating the method on real data of various kinds.

Too high a value may give too much weight to the prior hypothesis of spatial smoothness, and produce poorly clustered classes in the features space. For a Gaussian mixture model, satisfying image segmentations β is tuned based on what the user knew to be the desired output. In most applications it would be more practical to have some way to determine it automatically. The relationship with Markov random fields priors would suggest to use the techniques that were proposed in image segmentation to estimate the parameters of the multilevel logistic prior[2]. Most of these techniques require however computationally intensive Monte-Carlo simulations[11][56].

Future work could focus in a likelihood-based heuristic consisting of using the algorithm with different values of β, $\{\beta_1, \ldots, \beta_q\}$, producing thus different parameter estimates $\{\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_q\}$; plotting the log-likelihoods of the mixture parameters $L(\boldsymbol{\gamma}_j)$ against β_j could then suggest to retain the highest possible β that yields a reasonably high $L(\boldsymbol{\gamma})$. This assumes that a good value for β should yield a partition that is close to the 'true' classification, and that such a partition should give a parameter estimate that is close to the parameters that globally maximize the log-likelihood function; conversely, an excessive spatial smoothing is supposed to yield a partition for which parameter estimates have a relatively low log-likelihood.

The clustering result of algorithm also largely depends on $\boldsymbol{\alpha}$, since depending on the initial pixels membership very different segmentation results can be obtained. A reasonable solutions to this could be to implement some kind of pre-processing training stage where the $\boldsymbol{\alpha}$ matrix should be estimated according to the testing images.

In addition to this, in order to improve the performance of the method, it would also be very interesting to find and study other unsupervised clustering validation measures, a more accurate ones. In that way, it would be easier to test the 'goodness' of the algorithm in more realistic unsupervised data.

Lastly, regarding to the CPD project, its information system contains information about incidents, victims, offenders, gangs, calls for service, suspicious activity, critical facilities, community concerns, and more. Process these data and work with them would be of great interest. Besides, thinking in a possible incorporation of tools to analyze trends in variables that might not be traditional police information sources, such as weather, economic data, demographics, reported community concerns, troubled buildings, and penitentiary releases, could also be helpful to predict future crime and resource allocation needs.

There is still a long way to run.

Appendix A

Implemetation Remarks

This project is completely developed in Matlab, using its Toolbox libraries.

Even though we have created all the simulated images, implemented all the segmentations methods, and conducted all the experiments with Matlab, in this appendix only the code corresponding to the proposed ‘Spatial Fuzzy Clustering using Markov Random Fields’ method is included; since for the other methods, although slightly different, there are already written Matlab codes, and the codes corresponding to the computer simulated images and the experiments have been considered not as interesting, they are not included.

However not included, we must emphasize the complexity and usefulness of the Mapping Toolbox. We used it in order to work with the data provided by the *Chicago Police Department* (CPD). As it is said in Chapter 7, future work related to the CPD project will be mainly developed with the help of this Toolbox.

With the intention of developing a modular code, the code has been divided into different functions, trying to be as general as possible. With this, we want to get a solution where some of the parts of the algorithm can be changed, for example using another method different from the CGM in order to solve the optimization procedure, but keeping the general structure of the algorithm.
On the following pages the different functions codes are shown.

A.1 Main Function

This function contains, explicitly or implicitly, all the other functions. Providing the initial parameters, this is to say: the original image, the region that we want to segment of this image (in our case it is the whole image), the regularization parameter and the number of clusters, it provides the final segmentation and plots it.

```
function [ labels,alpha ] = MRFkclus( im,mask,betaparam,K )
%main program, cluster assignement/image segmentation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% im            MxNxD       image
%               MxN : number of pixels
%               D:    number of spectrums
% betaparam                 regularization term (trade-off between the cost
%                           and the penalty terms, i.e. feature and spatial
%                           information)
% mask          MxN         region of interest
% K                         number of clusters/different segmentation
%                           regions
% alpha         MxNxKxb     degree of membership of each pixel to each class
%               b           number of different beta
%               MxN : number of pixels
% labels        MxN         labeled image
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[M,N,D]=size(im);
mask=reshape(mask,M,N);
[row,col]=find(mask);
%edges of the region of interest
col_min=min(col);
col_max=max(col);
row_min=min(row);
row_max=max(row);
betaparam=betaparam/100;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
X=zeros(size(mask));
for d=1:D
    x=im(:,:,d);
    x=reshape(x,M,N);
    x(isnan(x))=0;
    x=x.*mask;
    X(:,:,d)=x;
end
%plotting the original image
figure
for d=1:D
        subplot(1,D,d)
        imagesc(im(:,:,d));
        title(['spectrum' int2str(d)])
end
%%%%%%%%%%%%%%%%%%%%%% classification %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for b=1:size(betaparam,2)
            [alpha(:,:,:,b),C]=MRF_cgsm(X,mask,betaparam(b),K);
    for m=1:M
        for n=1:N
         [maxim,l]=max(alpha(m,n,:,b));
         labels(m,n,b)=l(1);
        end
    end
%plotting the segmentation result
    figure
    imagesc(labels(:,:,b))
    colormap(jet)
    colorbar
    title('Labeled Image')
end
end
```

A.2 Conjugate Gradient Method

The role of this function is to compute the *Conjugate Gradient Method* in order to get the solution of the optimization problem, which, eventually, provides the final segmentation.

```
function [ alpha ,C] = MRF_cgsm( im, mask, b, K )
% computing the Conjugate Gradient Method in order to find the optimal alpha
% (segmentation)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% K                            number of clusters
% im            MxNxD          image
%               D:    number of spectrums
% b                            regularization term (trade-off between the cost
%                              and the penalty terms, i.e. feature and spatial
%                              information)
% mask          MxN            region of interest
% alpha         MxNxK          degree of membership of each pixel to each class
%               MxN : number of pixels
% C                            cost function at each iteration (until
%                              convergence)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[M,N,D]=size(im);
%%%%%%%%%%%%%%%%%%%%%%%%%%%% alpha initialization %%%%%%%%%%%%%%%%%%%%%%%%%%%%
 for m=1:M
    for n=1:N
        for k=1:K
            alpha(m,n,k)=rand;
        end
        alpha(m,n,:)=alpha(m,n,:)/sum(alpha(m,n,:));
    end
 end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
inialph=alpha;
iterations=max(M,N)+10;
betaparam=0;
    for it=1:iterations
        auxim=im;
        if it>1          %first iteration without regularization
            betaparam=b;
        end
        C(it)=costFunctionKclasses(alpha, auxim,betaparam,mask);
        auxC=C(it);
        newC=auxC;
        grad=CGMgrad(alpha,auxim,betaparam,mask);
        for k=1:K
            grad=real(grad);
            g(:,k)=reshape(grad(:,:,k),[],1);
            gamma(k)=1e-6/mean(abs(g(k)));
            newalpha(:,:,k)=alpha(:,:,k)-gamma(k).*grad(:,:,k);
        end
        z=0;zz=0;
        while newC<= auxC && z<iterations
            z=z+1;

            oldC=newC;
            for k=1:K
                gamma(k)=2*gamma(k);
                oldalpha(:,:,k)=newalpha(:,:,k);
                newalpha(:,:,k)=alpha(:,:,k)-gamma(k).*grad(:,:,k);
                %avoid zig-zag effect
                if min(reshape(alpha(:,:,k),[],1)) < -0.01
                    z=iterations*2;
                end
                if max(reshape(alpha(:,:,k),[],1)) >1
                    z=iterations*2;
                end
            end
            newC=costFunctionKclasses(newalpha,auxim,betaparam,mask);
        end
        alpha=oldalpha;
```

```
            C(it)=costFunctionKclasses(alpha, auxim,betaparam,mask);
      end
end
```

A.3 Cost Function

This Matlab code gets the value of the cost function when the parameters are the given ones. This function is used when computing the CGM.

```matlab
function [ cost ] = costFunctionKclasses( alpha, im, betaparam, mask )
% computing the value of the cost function with the given parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% alpha         MxNxK       degree of membership of each pixel to each class
%               MxN : number of pixels
%               K :   number of clusters
% im            MxNxD       image
%               D:    number of spectrums
% betaparam                 regularization term (trade-off between the cost
%                           and the penalty terms, i.e. feature and spatial
%                           information)
% mask          MxN         region of interest
% cost                      value of the cost function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    [M,N,K]=size(alpha);
    D=size(im,3);
    for d=1:D
        suma(:,:,d)=zeros(M,N);
    end
    for d=1:D
        for k=1:K
            Num(k)=sum(reshape(alpha(:,:,k),[],1));
            % mean and cov estimation for each cluster
            mu(d,k)=(1/Num(k))*sum(sum(mask.*alpha(:,:,k).*im(:,:,d)));
            sigmasq(d,k)=(1/Num(k))*sum(sum(mask.*alpha(:,:,k).*...
                (im(:,:,d)-mu(d,k)).^2));
            if k~=K
                c(:,:,k,d)=alpha(:,:,k).*((1./sqrt(2*pi*sigmasq(d,k)))*...
                    exp(-(im(:,:,d)-mu(d,k)).^2./(2*sigmasq(d,k)))).*mask;
                suma(:,:,d)=suma(:,:,d)+alpha(:,:,k);
                neighbour(:,:,1)=circshift(alpha(:,:,k),[1 0]);
                neighbour(:,:,2)=circshift(alpha(:,:,k),[-1 0]);
                neighbour(:,:,3)=circshift(alpha(:,:,k),[0 1]);
                neighbour(:,:,4)=circshift(alpha(:,:,k),[0 -1]);
                neighbour(:,:,5)=circshift(alpha(:,:,k),[1 1]);
                neighbour(:,:,6)=circshift(alpha(:,:,k),[-1 -1]);
                neighbour(:,:,7)=circshift(alpha(:,:,k),[-1 1]);
                neighbour(:,:,8)=circshift(alpha(:,:,k),[1 -1]);
                auxpenalty(:,:,k)=betaparam*((abs(alpha(:,:,k)-...
                    neighbour(:,:,1))).^2+(abs(alpha(:,:,k)-...
                    neighbour(:,:,2))).^2+(abs(alpha(:,:,k)-...
                    neighbour(:,:,3))).^2+(abs(alpha(:,:,k)-...
                    neighbour(:,:,4))).^2+...
                +(abs(alpha(:,:,k)-neighbour(:,:,5))).^2+...
                (abs(alpha(:,:,k)-neighbour(:,:,6))).^2+...
                (abs(alpha(:,:,k)-neighbour(:,:,7))).^2+...
                (abs(alpha(:,:,k)-neighbour(:,:,8))).^2);
            else
                c(:,:,k,d)=(1-suma(:,:,d)).*((1./...
                    sqrt(2*pi*sigmasq(d,k)))*exp(-(im(:,:,d)-...
                    mu(d,k)).^2./(2*sigmasq(d,k)))).*mask;
            end
        end
        penalty=sum(auxpenalty,3);
        auxcost(:,:,d)=log(sum(c(:,:,:,d),3))-penalty;
    end
    cost=-sum(sum(sum(auxcost)));
end
```

A.4 Gradient Function

Eventually, this function is used for calculating the gradient map of the image. This result is also used during the CGM computation.

```
function [grad] = CGMgrad( alpha, im, betaparam, mask )
% in this function, the gratient map 'grad' is computed.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% alpha         MxNxK       degree of membership of each pixel to each class
%               MxN : number of pixels
%               K :   number of clusters
% im            MxNxD       image
%               D:    number of spectrums
% betaparam                 regularization term (trade-off between the cost
%                           and the penalty terms, i.e. feature and spatial
%                           information)
% mask          MxN         region of interest
% grad          MxNxK       gradient matrix for each alpha
%                           (cluster assignment)
% penalty       MxNxK       penalty term, for each pixel for each cluster
%                           assignment. A clique of 4x4 pixels is
%                           considered, i.e. the 8 nearest neighbors.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[M,N,K]=size(alpha);
    D=size(im,3);
    for d=1:D
        for k=1:K
            Num(k)=sum(reshape(alpha(:,:,k),[],1));
            % mean and cov estimation for each cluster and their derivates
            mu(d,k)=(1/Num(k))*sum(sum(mask.*alpha(:,:,k).*im(:,:,d)));
            sigmasq(d,k)=(1/Num(k))*sum(sum(mask.*alpha(:,:,k).*...
                (im(:,:,d)-mu(d,k)).^2));
            sigma(d,k)=sqrt(sigmasq(d,k));
            dervm(:,:,d,k)=(im(:,:,d)/Num(k)-mu(d,k)/Num(k)).*mask;
            dervsigma(:,:,d,k)=(((im(:,:,d)-mu(d,k)).^2-2*alpha(:,:,k).*...
                (im(:,:,d)-mu(d,k))*dervm(d,k))/2/Num(k)/sigma(d,k)-...
                sigma(d,k)/2/Num(k)).*mask;
        end
    end
    for k=1:K
        grad(:,:,k)=zeros(M,N);
    end
%computing penalty term, clique of 4x4 pixels
    for k=1:(K-1)
        neighbour(:,:,1)=circshift(alpha(:,:,k),[1 0]);
        neighbour(:,:,2)=circshift(alpha(:,:,k),[-1 0]);
        neighbour(:,:,3)=circshift(alpha(:,:,k),[0 1]);
        neighbour(:,:,4)=circshift(alpha(:,:,k),[0 -1]);
        neighbour(:,:,5)=circshift(alpha(:,:,k),[1 1]);
        neighbour(:,:,6)=circshift(alpha(:,:,k),[-1 -1]);
        neighbour(:,:,7)=circshift(alpha(:,:,k),[-1 1]);
        neighbour(:,:,8)=circshift(alpha(:,:,k),[1 -1]);
        penalty(:,:,k)=2*betaparam*(abs(alpha(:,:,k)-neighbour(:,:,1))+...
            abs(alpha(:,:,k)-neighbour(:,:,2))+...
            +abs(alpha(:,:,k)-neighbour(:,:,3))+abs(alpha(:,:,k)-...
            neighbour(:,:,4))+abs(alpha(:,:,k)-neighbour(:,:,5))+...
            abs(alpha(:,:,k)-neighbour(:,:,6))+...
            +abs(alpha(:,:,k)-neighbour(:,:,7))+abs(alpha(:,:,k)-...
            neighbour(:,:,7))).*((alpha(:,:,k)>neighbour(:,:,1))-...
            -(alpha(:,:,k)<=neighbour(:,:,1))+(alpha(:,:,k)>...
            neighbour(:,:,2))-(alpha(:,:,k)<=neighbour(:,:,2))+...
            +(alpha(:,:,k)>neighbour(:,:,3))-(alpha(:,:,k)<=...
            neighbour(:,:,3))+(alpha(:,:,k)>neighbour(:,:,4))-...
            (alpha(:,:,k)<=neighbour(:,:,4))+...
            +(alpha(:,:,k)>neighbour(:,:,5))-...
            -(alpha(:,:,k)<=neighbour(:,:,5))+(alpha(:,:,k)>...
            neighbour(:,:,6))-(alpha(:,:,k)<=neighbour(:,:,6))+...
            +(alpha(:,:,k)>neighbour(:,:,7))-(alpha(:,:,k)<=...
            neighbour(:,:,7))+(alpha(:,:,k)>neighbour(:,:,8))-...
            (alpha(:,:,k)<=neighbour(:,:,8)));
    end
    penalty(:,:,K)=0;
    for d=1:D
        suma(:,:,d)=zeros(M,N);
    end
%computing gradient
    for m=1:M
        for n=1:N
            for d=1:D
                for k=1:K
                    if k~=K
                        t1(d,k)=(1/sqrt(2*pi*sigmasq(d,k)))*...
                            exp(-(im(m,n,d)-mu(d,k))^2/(2*sigmasq(d,k)));
                        t2(d,k)=alpha(m,n,k)*exp(-(im(m,n,d)-mu(d,k))^2/...
                            (2*sigmasq(d,k)))/sqrt(2*pi)/(-sigmasq(d,k))...
                            *dervsigma(m,n,d,k);
                        t31(d,k)=(dervm(m,n,d,k)*(im(m,n,d)-mu(d,k)))/...
                            sigmasq(d,k)+sigma(d,k)*(dervsigma(m,n,d,k)*...
                            (im(m,n,d)-mu(d,k))^2)/(sigmasq(d,k)^2);
                        t3(d,k)=alpha(m,n,k)*exp(-(im(m,n,d)-mu(d,k))^2/...
                            (2*sigmasq(d,k)))/sqrt(2*pi*sigmasq(d,k))*...
                            t31(d,k);
                        t5(d,k)=alpha(m,n,k)*((1./sqrt(2*pi*...
                            sigmasq(d,k)))*exp(-(im(m,n,d)-mu(d,k))^2./...
                            (2*sigmasq(d,k))));
                        t4(d,k)=-(1/sqrt(2*pi*sigmasq(d,K)))*...
                            exp(-(im(m,n,d)-mu(d,K))^2/2/sigmasq(d,K));
                        suma(m,n,d)=suma(m,n,d)+alpha(m,n,k);
                    else
                        t1(d,k)=0;
                        t2(d,k)=(1-suma(m,n,d))*...
                            exp(-(im(m,n,d)-mu(d,k))^2/(2*sigmasq(d,k)))...
                            /sqrt(2*pi)/(-sigmasq(d,k))*dervsigma(m,n,d,k);
                        t31(d,k)=(dervm(m,n,d,k)*(im(m,n,d)-mu(d,k)))/...
                            sigmasq(d,k)+sigma(d,k)*(dervsigma(m,n,d,k)*...
                            (im(m,n,d)-mu(d,k))^2)/(sigmasq(d,k)^2);
                        t3(d,k)=(1-suma(m,n,d))*...
                            exp(-(im(m,n,d)-mu(d,k))^2/(2*sigmasq(d,k)))...
                            /sqrt(2*pi*sigmasq(d,k))*t31(d,k);
                        t4(d,k)=0;
                        t5(d,k)=(1-suma(m,n,d))*...
                            ((1./sqrt(2*pi*sigmasq(d,k)))*...
                            exp(-(im(m,n,d)-mu(d,k))^2./(2*sigmasq(d,k))));
                    end
                end
            end
            t1(isnan(t1))=0;
            t2(isnan(t2))=0;
            t3(isnan(t3))=0;
            t4(isnan(t4))=0;
            t5(isnan(t5))=0;
            auxgrad=sum((t1+t2+t3+t4)./t5);
            for k=1:K
                grad(m,n,k)=auxgrad(k)-penalty(m,n,k);
                grad(m,n,k)=- grad(m,n,k);
            end
        end
    end

end
```

References

[1] M. N. Ahmend, S.M. Yamany, N. Mohamed, A. A. Farag, T. Moriarty. *A modified fuzzy c-means algorithm for bias field estimation and segmentation of MRI data.* IEEE Trans. on Medical Imaging 21, pp. 193-199, 2002.

[2] C. Ambroise. *Approche probabiliste en classifiacation automatique et contraintes de voisinage.* PhD thesis, Université de Technologie de Compiègne, 1996.

[3] C. Ambroise, G. Govaert. *Convergence of an EM-type algorithm for spatial clustering.* Pattern Recognition Letters 19, pp. 919-927, 1998.

[4] M. Bertero, T.A. Poggio,V. Torre. *Ill-posed problems in early vision.* Proceedings of the IEEE, 76(8):869–889, 1988.

[5] Jacob Bank, Benjamin Cole. *Calculating the Jaccard Similarity Coefficient with Map Reduce for Entity Pairs in Wikipedia.* December 16, 2008.

[6] J. Besag. *Spatial interaction and the statistical analysis of lattice system.* Journal of the Royal Statistical Society Series B 36, pp. 192-236, 1974.

[7] J. Besag. *On the statistical analysis of dirty pictures.* Journal of the Royal Statistical Society Series B 48, pp. 259-302, 1986.

[8] J. Besag. *Efficiency of pseudolikelihood estimation for simple Gaussian fields.* Biometrica 64, pp. 616-618, 1977.

[9] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* New York, Plenum Press, 1981.

[10] Noel Black, Shirley Moore. *Conjugate Gradient Squared Method.* MathWorld–A Wolfram Web Resource, created by Eric W. Weisstein, 2010.

[11] B. Chalmond. *An iterative Gibbsian technique for reconstruction of m–ary images.* Pattern recognition 22(6), pp. 747-761, 1989.

[12] J. J. Clark, A. L. Yuille. *Data Fusion for Sensory Information Processing Systems.* Kluwer Academic Publishers, Norwell, MA, 1990.

[13] M. Dang, G. Govaert. *Spatial Fuzzy Clustering using EM and Markov Random Fields.* System Research and Information Systems, Vol. 8, pp. 183-202, 1998.

[14] A. P. Dempster, N. M. Laird, D. B. Rubin. *Maximum Likelihood from Incomplete Data via the EM Algorithm.* Journal of the Royal Statistical Society. Series B (Methodological), Vol. 39, No. 1. , pp. 1-38, 1977.

[15] Huawu Deng, David A. Clausi. *Unsupervised image segmentation using a simple MRF model with a new implementation scheme.* Department of Systems Design Engineering, University of Waterloo, Canada, 2004.

[16] H. Derin, W. S. Cole. *Segmentation of textured images using Gibbs random fields.* Computer Vision, Graphics and Image Processing, 35:72–98, 1986.

[17] H. Derin, H. Elliot. *Modeling and segmentation of noisy and textured images using Gibbs random fields.* IEEE Transactions on Pattern Analysis and Machine Intelligence 9, pp. 39-55, 1987.

[18] I. M. Elfadel. *From random fields to networks.* PhD thesis, MIT, Cambridge, MA, USA, 1993.

[19] Davide Eynard. *Methods for Intelligent Systems: Lecture Notes on Clustering*, Chapter 6. Department of Electronics and Information, Politecnico di Milano, 2010.

[20] Yanqiu Feng, Wufan Chen. *Brain MR Image segmentation Using Fuzzy Clustering with Spatial Constraints Based on Markov Random field Theory* . Laboratory of Medical Imaging, Department of Biomedical Engineering, First Military Medical University, Guangzhou, P.R China, 2004.

[21] S. Geman, D. Geman. *Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images.* IEEE Trans. Pattern Anal. Mach. Intell. 6 (6), pp. 721-741, 1984.

[22] Ricardo Gutierrez-Osuna. *Introduction to Pattern Analysis.* Texas A&M University, 2010.

[23] J. M. Hammersley$^{(1)}$, P. Clifford$^{(2)}$. *Markov Fields on Finite Graphs and Lattices.* (1) Univeristy of California, Berkeley and Oxford University, (2) Univeristy of California, Berkeley and Bristol University, 1971.

[24] Jiawei Han, Micheline Kamber. *Data mining: concepts and techniques.* Morgan Kaufmann, 2006

[25] Trebor Hastie, Robert Tibshirani, Jerome H. Friedman. *The Elements of Statistical Learning.* Springer Verlag, 2001.

[26] R. Hathaway. *Another inerpretation of the EM algorithm for mixture distributions.* Journal of Statistics & Probability Letters 4, pp. 53-56, 1986.

[27] Lili He, Ian R. Greenshields. *Biomedical Signal Processing and Control: An MRF spatial fuzzy clustering method for fMRI SPMs.* University of Connecticut, Computer Science and Engineering, August 2008.

[28] E. Jaynes. *On the rationale of maximum-entropy methods.* Proceedings of the IEEE, 70(9):939–952, 1982.

[29] K. Kanatani. *Geometric computation for machine vision.* Oxford University Press, New York, 1993.

[30] Zoltan Kato. *Image Segmentation and Parameter Estimation in a Markovian Framework.* Shool of Computing, National University of Singapore, 2000.

[31] R. Kindermann, J. L. Snell. *Markov Random Fields and Their Applications.* American Mathematical Society, Providence, R.I., 1980.

[32] J. J. Koenderink. *Solid Shape.* MIT Press, 1990.

[33] Ludmila I. Kuncheva. *Combining pattern classifiers: methods and algorithms.* John Wiley & Sons, 2004.

[34] S. Z. Li. *Markov Random Field Modeling in Image Analysis.* 2_{nd} Ed., Springer-Verlag, 2001.

[35] X. Li, l. Li, H. Lu, D. Chen, Z. Liang. *Inhomogeneity correction for magnetic resonance images with fuzzy c-means algorithm.* Proc. SPIE 5032, pp.995-1005, 2003.

[36] A. W. C Liew, S.H. Leung, W.H. Lau. *Fuzzy image clustering incorporating spatial continuity.* IEE Proc. Visual Image Signal Process. 147, pp.185-192, 2000.

[37] Xin Liu, Deanna L. Langer, Masoom A. Haider, Yogui Yang, Miles N. Wernick, and İmam Şamil Yetik. *Prostate Cancer Segmentation with Simultaneous Estimation of Markov Random Field Parameters and Class.* IEEE Transactions on Medical Imaging, Vol.28. No.6., June 2009.

[38] Chang-tien Lu. *Spatial Clustering Methods in Data Mining.* 2005

[39] X. Lu, F. Zhou, J. Zhou. *Synthetic aperture radar image segmentation based on improved fuzzy Markov random field model.* 1_st Int. Symp. Sist. Control Aerospace Auronautics, pp. 1205-1208, Jan. 2006.

[40] J. L. Marroquin. *Probabilistic Solution of Inverse Problems.* PhD thesis, MIT AI Lab, 1985.

[41] J. Marroquin, S. Mitter, T. Poggio, T. *Probabilistic solution of ill-posed problems in computational vision.* Journal of the American Statistical Association, 82(397):76–89, 1987.

[42] Matteo Matteucci. *A Tutorial on Clustering Algorithms.* Politecnico di Milano, 2009.

[43] J. Moussouris. *Gibbs and Markov systems with constraints.* Journal of statistical physics, 10:11–33, 1974.

[44] J. L. Mundy, A. Zisserman. *Geometric Invariants in Computer Vision.* MIT Press, Cambridge, MA, 1992.

[45] S. G. Nadabar, A. K. Jain. *Parameter estimation in MRF line process model.* Proceedings of IEEE Computer society Conference on Computer Vision and Pattern Recognition, pp. 528-533, 1992.

[46] J.C. Noordam, W. H. A. M. van der Broek, L. M. C. Buydens. *Geometrically guided fuzzy C-meas clustering for multivariate image segmentation.* Proc. Int. Conf. on Pattern Recognition , pp. 462-465, 2000.

[47] T. Poggio, V. Torre, C. Koch. *Computational vision and regularization theory.* Nature, 317:314–319, 1985.

[48] S. Ruan, D. Bloyet, M. Revenu. *Segmentation of magnetic resonance images using Markov random fields.* Proc. Int. Conf. Image Process., vol.3, pp. 1051-1054, Oct. 2001.

[49] F. Salzenstein, W. Pieczynski. *PArameter estimation in hidden fuzzy Markov random fields and image segmentation.* Graphical Models Image Process., vol.59, no. 4, pp. 205-231, 1997.

[50] Jonathan Richard Shewchuk. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain.* 1st Ed, School of Computer Science Carnegie Mellon University, Pittsburgh, 1994.

[51] D. Stewart, D. Blacknell, A. Blake, R. Cook, C. Oliver. *Optimal approach to SAR image segmentation and classification.* IEEE Proc. Radar, Sonar Navigation 147 (3), pp. 134-142, 2000.

[52] Sergios Theodoridis, Konstantions Koutroumbas.*Pattern Recognition.* Academic Press, 1999.

[53] Sergios Theodoridis, Konstantions Koutroumbas.*Pattern Recognition 4th* Ed. Elsevier Inc., 2009.

[54] Y. A. Tolias, S.M. Panas. *On applying spatial constraints in fuzzy image clustering using fuzzy rule-based system.* IEEE Signal Processing Letters 5, pp. 245-247, 1998.

[55] Y.Yang, Ch. Zheng, P. Lin. *Fuzzy c-means clustering algorithm with a novel penalty term for image segmentation.* Key Laboratory of Biomedical Engineering, Xi'an Jiaotong University, China, 2005.

[56] L. Younes. *Parametric inference for imperfectly observed Gibbsian fields.* Probability Theory and Related Fields 82, pp. 625-645, 1989.

[57] L.A. Zadeh. *Fuzzy sets.* Inform. and Control 8, pp. 338-353, 1965.

[58] Osmar R. Zaane. *Principles of Knowledge Discovery in Database.* University of Alberta, 1999. Chapter 8.

[59] J. Zhang. *The mean field theory in EM procedures for Markov random fields.* IEEE Transactions on Image Processing 40, pp. 2570-2583, 1992.

[60] J. Zhang. *The mean field theory in EM procedures for blind Markov random field image restoration.* IEEE Transactions on Image Processing 2, pp. 27-40, 1993.

[61] *Segmentation Validation Engine.* Laboratory of Neuro Imaging UCLA, 2008.