The Pennsylvania State University

The Graduate School

Department of Computer Science and Engineering

A MACHINE LEARNING APPROACH TO CONTENT-BASED IMAGE INDEXING AND RETRIEVAL

A Thesis in

Computer Science and Engineering

by

Yixin Chen

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2003

We approve the thesis of Yixin Chen.

Date of Signature

James Z. Wang
Assistant Professor of Information Sciences and Technology
Thesis Adviser
Chair of Committee

Raj Acharya
Professor of Computer Science and Engineering
Chairman, Department of Computer Science and Engineering

C. Lee Giles
Professor of Information Sciences and Technology

Jia Li
Assistant Professor of Statistics

Donald Richards
Professor of Statistics

John Yen
Professor of Information Sciences and Technology

Abstract

In various application domains such as entertainment, biomedicine, commerce, education, and crime prevention, the volume of digital data archives is growing rapidly. The very large repository of digital information raises challenging problems in retrieval and various other information manipulation tasks. Content-based image retrieval (CBIR) is aimed at efficient retrieval of relevant images from large image databases based on automatically derived imagery features. However, images with high feature similarities to the query image may be very different from the query in terms of semantics. This discrepancy between low-level content features (such as color, texture, and shape) and high-level semantic concepts (such as sunset, flowers, outdoor scene, etc.) is known as "semantic gap," which is an open challenging problem in current CBIR systems.

With the ultimate goal of narrowing the semantic gap, this thesis makes three contributions to the field of CBIR. The first contribution is a novel region-based image similarity measure. An image is represented by a set of segmented regions each of which is characterized by a fuzzy feature (fuzzy set) reflecting color, texture, and shape properties. Fuzzy features naturally characterize the gradual transition between regions (blurry boundaries) within an image, and incorporate the segmentation-related uncertainties into the retrieval algorithm. The resemblance of two images is defined as the overall similarity between two families of fuzzy features, and quantified by a similarity measure that integrates properties of all the regions in the images. Compared with similarity measures based on individual regions and on all regions with crisp-valued

feature representations, the proposed measure greatly reduces the influence of inaccurate segmentation, and provides a very intuitive quantification.

The second contribution is a novel image retrieval scheme using unsupervised learning. It is built on a hypothesis that images of the same semantics tend to be clustered in some feature space. The proposed method attempts to capture semantic concepts by learning the way that images of the same semantics are similar and retrieving image clusters instead of a set of ordered images. Clustering is dynamic. In particular, clusters formed depend on which images are retrieved in response to the query. Therefore, the clusters give the algorithm as well as the users semantic relevant clues as to where to navigate. The proposed retrieval scheme is a general approach that can be combined with any real-valued symmetric similarity measure (metric or nonmetric). Thus it may be embedded in many current CBIR systems.

The last contribution is a novel region-based image classification method. An image is represented as a set of regions obtained from image segmentation. It is assumed that the concept underlying an image category is related to the occurrence of regions of certain types, which are called *region prototypes* (RPs), in an image. Each RP represents a class of regions that is more likely to appear in images with the specific label than in the other images, and is found according to an objective function measuring a co-occurrence of similar regions from different images with the same label. An image classifier is then defined by a set of rules associating the appearance of RPs in an image with image labels. The learning of such classifiers is formulated as a Support Vector Machine (SVM) learning problem with a special class of kernels.

Table of Contents

List of Tables

List of Figures

Acknowledgments

This work would not have been possible without the support from my sister, Yiling Chen, and my parents. As always, the greatest debt I owe is to them.

I am deeply indebted to my thesis adviser Professor James Z. Wang whose help, stimulating suggestions, and encouragement helped me in all the time of research for and writing of this thesis. James gave me the freedom to explore a variety of topics. Whenever I struggled, his generous support always came at just the right time.

I would like to thank Professors Lee Giles, Jia Li, Donald Richards, and John Yen, who have served on my thesis committee, for spending much time on reading this thesis and providing insightful commentary on my work. I am grateful to Professor Jia Li; she carefully read several of my papers, and her suggestions greatly improved the quality of the papers and the thesis.

I would also like to thank my colleagues Jinbo Bi, Ya Zhang, Xiang Ji, and Hui Han. Discussions with them at different stages of the thesis were very rewarding. Special thanks are also due to Dr. Robert Krovetz; our collaboration in the summer of 2002 paid off in the development of CLUE.

This research was supported by the National Science Foundation under Grant No. IIS-0219272, The Pennsylvania State University, the PNC Foundation, and by SUN Microsystems under Grant EDUD-7824-010456-US. Parts of this work were completed when I was supported by a summer internship at the NEC Research Institute. The source code of SIMPLIcity system (written by Jia Li and James Z. Wang) and the

software SVM^{Light} (written by Thorsten Joachims) helped in the development. Some of the materials in the thesis have been published in conferences and Transactions of IEEE and ACM [15, 16, 17, 18, 19, 20].

Chapter 1

Introduction

With the rapid growth of the Internet and the falling price of storage devices, it becomes increasingly popular to store texts, images, graphics, video, and audio in digital format. This raises the challenging problem of designing techniques that support effective search and navigation through the contents of large digital archives. As part of this general problem, image retrieval and indexing have been active research areas for more than a decade.

Content-based image retrieval (CBIR) is aimed at efficient retrieval of relevant images from large image databases based on automatically derived imagery features. These features are typically extracted from shape, texture, or color properties of query image and images in the database. Potential applications include digital libraries, commerce, Web searching, geographic information systems, biomedicine, surveillance and sensor systems, commerce, education, crime prevention, etc.

This thesis makes three contributions that are closely related to CBIR. The first contribution is a novel region-based image similarity measure. This measure greatly increases the robustness of the retrieval system against segmentation-related uncertainties. The second contribution is a novel image retrieval scheme using unsupervised learning. It retrieves image clusters based not only on the feature similarity of images to the query,

but also on how images are similar to each other. The last contribution is a novel image categorization algorithm that classifies images based on the information of regions contained in the images. The concept underlying an image category is related to the occurrence of regions of certain properties. Such a relationship is captured by a set of rules obtained from learning.

The remainder of the thesis is organized as follows:

- **Chapter 2. Related Work in Image Retrieval and Categorization**

  Content-based image retrieval (CBIR) and image categorization are two closely related and rapidly expanding research areas. CBIR aims at developing techniques that support effective searching and browsing of large image digital libraries based on automatically derived image features. Image categorization refers to classifying images into a collection of predefined categories. We review the related work in CBIR and image categorization.

- **Chapter 3. Related Work in Machine Learning**

  The field of machine learning is concerned with constructing computer programs that automatically improve with experience. Machine learning draws on concepts and results from many fields, including artificial intelligence, statistics, control theory, cognitive science, and information theory. In this chapter, we summarize three well-known techniques, Support Vector Machine (SVM), additive fuzzy systems, and spectral graph clustering, which will be used in this thesis.

- **Chapter 4. Support Vector Learning for Fuzzy Rule-Based Classification Systems**

To design a fuzzy rule-based classification system (fuzzy classifier) with good generalization ability in a high dimensional feature space has been an active research topic for a long time. As a powerful machine learning approach for pattern recognition problems, support vector machine (SVM) is known to have good generalization ability. More importantly, an SVM can work very well on a high (or even infinite) dimensional feature space. In this chapter, we investigate the connection between fuzzy classifiers and kernel machines, establish a link between fuzzy rules and kernels, and propose a learning algorithm for fuzzy classifiers. The result will be used in Chapter 7.

- **Chapter 5. A Robust Image Similarity Measure Using Fuzzified Region Features**

  This chapter proposes a fuzzy logic approach, UFM (unified feature matching), for region-based image retrieval. In our retrieval system, an image is represented by a set of segmented regions each of which is characterized by a fuzzy feature (fuzzy set) reflecting color, texture, and shape properties. As a result, an image is associated with a family of fuzzy features corresponding to regions. Fuzzy features naturally characterize the gradual transition between regions (blurry boundaries) within an image, and incorporate the segmentation-related uncertainties into the retrieval algorithm. The resemblance of two images is then defined as the overall similarity between two families of fuzzy features, and quantified by a similarity measure, UFM measure, which integrates properties of all the regions in the images. Compared

with similarity measures based on individual regions and on all regions with crisp-valued feature representations, the UFM measure greatly reduces the influence of inaccurate segmentation, and provides a very intuitive quantification.

- **Chapter 6. Cluster-Based Retrieval of Images by Unsupervised Learning**

  In a typical content-based image retrieval (CBIR) system, query results are a set of images sorted by feature similarities with respect to the query. However, images with high feature similarities to the query may be very different from the query in terms of semantics. This discrepancy between low-level features and high-level concepts is known as the semantic gap. This chapter introduces a novel image retrieval scheme, CLUster-based rEtrieval of images by unsupervised learning (CLUE), which attempts to tackle the semantic gap problem based on a hypothesis that *images of the same semantics are similar in a way, images of different semantics are different in their own ways*. CLUE attempts to capture semantic concepts by learning the way that images of the same semantics are similar and retrieving image clusters instead of a set of ordered images. Clustering in CLUE is dynamic. In particular, clusters formed depend on which images are retrieved in response to the query. Therefore, the clusters give the algorithm as well as the users semantic relevant *clues* as to where to navigate. CLUE is a general approach that can be combined with any real-valued, symmetric, metric or non-metric similarity measure, and thus it may be embedded in many current CBIR systems.

- **Chapter 7. Image Categorization by Learning and Reasoning with Regions**

  Designing computer programs that can automatically categorize images into a collection of predefined classes using low-level features is an important and challenging research topic in image analysis and computer vision. This chapter introduces a novel image categorization algorithm that classifies images based on the information of regions contained in the images. An image is represented as a set of regions obtained from image segmentation. It is assumed that the concept underlying an image category is related to the occurrence of certain types of regions, called *region prototypes*, in an image. Each region prototype represents a class of regions that are more likely to appear in images with the specific label than in remaining images, and is found according to an objective function measuring a co-occurrence of similar regions from different images with the same label. An image classifier is then defined by a set of rules associating the appearance of region prototypes in an image with image labels. The learning of such classifiers is formulated as a Support Vector Machine (SVM) learning problem with a special class of kernels. As a result, a collection of SVMs are trained, each corresponding to one image category.

- **Chapter 8**

  This chapter provides experimental evaluations of the three algorithms described in Chapter 5, Chapter 6, and Chapter 7. The performance is illustrated using examples from an image database of about $60,000$ general-purpose images. In

addition, images returned by Google's Image Search are used to demonstrate the potential of applying CLUE to real world image data and integrating CLUE as a part of the interface for keyword-based image retrieval systems.

- **Chapter 9. Conclusions and Future Work**

  In this chapter, we first summarize the contributions of this thesis on applying machine learning techniques to content-based image indexing and retrieval. Then we examine the limitations of the proposed approaches. Finally, we discuss some directions of future work.

Chapter 2

Related Work in Image Retrieval and Categorization

Both image retrieval and image categorization are active research areas. They are highly interdisciplinary areas situated at the intersection of databases, information retrieval, and computer vision. This chapter provides a brief review the relevant work.

2.1 Image Retrieval

Depending on the query formats, image retrieval algorithms roughly belong to two categories: keyword-based approaches and content-based methods. The keyword based approaches are based on the idea of storing a keyword (or keywords) description of the image content, created by a user on input, in addition to a pointer to the raw image data. Image retrieval is then shifted to standard database management capability combined with information retrieval techniques. Some commercial image search engines, such as Google Image Search and Lycos Multimedia Search, are keyword-based image retrieval systems.

Manual annotation for a large collection of images is not always available. Sometimes it may be extremely difficult to annotate an image using several keywords. This motivates research on content-based image retrieval (CBIR): retrieval of images by image example where a query image or sketch is given as input by a user. Generally speaking,

CBIR aims to develop techniques that support effective searching and browsing of large image digital libraries based on automatically derived image features.

In the past decade, many general-purpose image retrieval systems have been developed. Examples include IBM QBIC System [30], MIT Photobook System [79], Berkeley Chabot [77] and Blobworld Systems [10], Virage System [41], Columbia VisualSEEK and WebSEEK Systems [98], the PicHunter System [23], UCSB NeTra System [66], UIUC MARS System [71], the PicToSeek System [38], and Stanford WBIIS [121] and SIMPLIcity Systems [120], to name just a few.

From a computational perspective, a typical CBIR system views the query image and images in the database (target images) as a collection of features, and ranks the relevance between the query image and any target image in proportion to a similarity measure calculated from the features. In this sense, these features, or signatures of images, characterize the *content* of images. According to the scope of representation, features fall roughly into two categories: global features and local features. The former category includes texture histogram, color histogram, color layout of the whole image, and features selected from multidimensional discriminant analysis of a collection of images [30, 41, 79, 98, 105]. In the latter category are color, texture, and shape features for subimages [81], segmented regions [10, 16, 66, 120], and interest points [93].

As a key issue in CBIR, the similarity measure quantifies the resemblance in contents between a pair of images [89]. Depending on the type of features, the formulation of the similarity measure varies greatly. The Mahalanobis distance [42] and intersection distance [104] are commonly used to compute the difference between two histograms with the same number of bins. When the number of bins are different, the Earth Mover's

Distance (EMD) [87] applies. The EMD is computed by solving a linear programming problem. Moments [59], the Hausdorff metric [49], elastic matching [7], and decision trees [52] have been proposed for shape comparison. In [76], a similarity measure is defined from subjective experiments and multi-dimensional scaling (MDS) based upon the model of human perception of color patterns. Barnard et al. [5] presented a probability-based similarity measure that combines the information provided by text and the visual information provided by image features. The similarity measure in [12] assesses the topological relationships of image regions represented as a 2D string structure. Li et al. [64] presented an integrated region matching (IRM) scheme for region-based image retrieval. The IRM measure allows many-to-many region-based matching.

In one way or another, the aforementioned similarity measures capture certain facets of image content, named the *similarity-induced semantics*. Nonetheless, the meaning of an image is rarely self-evident. Similarity-induced semantics usually does not coincide with the high-level concept conveyed by an image (*semantics* of the image). This is referred to as the *semantic gap* [97], which reflects the discrepancy between the relatively limited descriptive power of low-level visual features (together with the associated similarity measure and the retrieval strategy) and high-level concepts.

Many approaches have been proposed to reduce the semantic gap. They generally fall into two classes depending on the degree of user involvement in the retrieval: relevance feedback and image database preprocessing using statistical classification. Relevance feedback is a powerful technique originally used in the traditional text-based information retrieval systems. In CBIR, a relevance-feedback-based approach allows a user to interact with the retrieval algorithm by providing information regarding the images

which the user believes to be relevant to the query [23, 73, 88]. Based on user feedback, the model of similarity measure is dynamically updated to give a better approximation of the perception subjectivity. There are also works that combine relevance feedback with supervised learning [110, 132]: binary classifiers are trained on-the-fly based on user feedback. Empirical results demonstrate the effectiveness of relevance feedback for certain applications. Nonetheless such a system may add burden to a user especially when more information is required than just Boolean feedback (relevant or non-relevant).

Statistical classification methods group images into semantically meaningful categories using low-level visual features so that semantically-adaptive searching methods applicable to each category can be applied [95, 112, 120, 63]. For example, the SemQuery system [95] categorizes images into different set of clusters based on their heterogeneous features. Vailaya et al. [112] organized vacation images into a hierarchical structure. At the top level, images are classified as indoor or outdoor. Outdoor images are then classified as city or landscape that is further divided into sunset, forest, and mountain classes. SIMPLIcity system [120] classifies images into graph, textured photograph, or non-textured photograph, and thus narrows down the searching space in a database. ALIP system [63] uses categorized images to train hundreds of two-dimensional multiresolution hidden Markov models each corresponding to a semantic category. Although these classification methods are successful in their specific domains of application, the simple ontologies built upon them could not incorporate the rich semantics of a sizable image database. There has been work on attaching words to images by associating the regions of an image with object names based on a statistic model [5]. But as noted by the authors in [5], the algorithm relies on semantically meaningful segmentation. And

semantically precise image segmentation by an algorithm is still an open problem in computer vision [96, 119, 133].

2.2 Image Categorization

The term image categorization refers to the labeling of images into one of a number of predefined categories. Although this is usually not a very difficult task for humans, it has proved to be an extremely difficult problem for machines (or computer programs). Major resources of difficulty include variable and sometimes uncontrolled imaging conditions, complex and hard-to-describe objects in an image, objects occluding other objects, and the gap between arrays of numbers representing physical images and conceptual information perceived by humans. Designing automatic image categorization algorithms has been an important research field for decades. Potential applications include digital libraries, Web searching, geographic information systems, biomedicine, surveillance and sensor systems, commerce, and education. In terms of CBIR, image categorization can be applied as a preprocessing stage: grouping images in the database into semantically meaningful categories. Within the areas of image processing, computer vision, and pattern recognition, there has been abundance of prior work on detecting, recognizing, and classifying a relatively small set of objects or concepts in specific domains of application [31, 101].

In Marr's classical book on computational and mathematical approach to vision [69], visual perception is described as "*the process of discovering from images what is present in the world and where it is*" ([69], p.3). Marr's characterization of vision emphasizes the process of extracting useful information from patterns perceived and

processing information to achieve conceptual clarity. This may also be viewed as a computational abstraction of most current image categorization methods, which only vary in algorithmic details, i.e., how information is extracted, represented, and processed.

As one of the simplest representations of digital images, histograms have been widely used for various image categorization problems. Szummer and Picard [106] used k-nearest neighbor classifier on color histograms to discriminate between *indoor* and *outdoor* images. In [112], Bayesian classifiers using color histograms and edge directions histograms are implemented to organize *sunset/forest/mountain* images and *city/landscape* images, respectively. Chapelle et al. [13] applied Support Vector Machines (SVMs) [9], which are built on color histograms, to classify images containing a generic set of objects. Although histograms can usually be computed with little cost and are effective for certain classification tasks, an important drawback of a global histogram representation is that information about spatial configuration is ignored. Many approaches have been proposed to tackle the drawback. In [48], a classification tree is constructed using color correlograms. Color correlogram captures the spatial correlation of colors in an image. Gdalyahu and Weinshall [33] applied local curve matching for shape silhouette classifications, in which objects in images are represented by their outlines.

A number of subimage-based methods have been proposed to utilize local and spatial properties by dividing an image into fixed-size blocks. In the method introduced by Gorkani and Picard [40], an image is first divided into 16 non-overlapping equal-sized blocks. Dominant orientations are computed for each block. The image is then classified as *city* or *suburb* scenes as determined by the majority orientations of blocks. Wang et

al. [120] developed a *graph/photograph* [1] classification algorithm. The classifier partitions an image into blocks and classifies every block into one of two categories based on wavelet coefficients in high frequency bands. If the percentage of blocks classified as photograph is higher than a threshold, the image is marked as photograph; otherwise, the image is marked as graph. Yu and Wolf [128] presented a one-dimensional Hidden Markov Model (HMM) for *indoor/outdoor* scene classification. The model is trained on vector quantized color histograms of image blocks. In ALIP system [63], a concept corresponding to a particular category of images is captured by a two-dimensional multiresolution HMM trained on color and texture features of image blocks. In this model, spatial relations among blocks and across image resolutions are both taken into consideration. Maron and Ratan [67] formulated image categorization into a Multiple-Instance Learning (MIL) problem [68]. Images are represented as collections of fixed-size, possibly overlapping, image patches. Simple templates are learned from patches to represent classes of natural scene images.

Although a rigid partition of an image into fixed-size blocks preserves certain spatial information, it often breaks an object into several blocks or puts different objects into a single block. Thus visual information about objects, which could be beneficial to image categorization, may be destroyed by a rigid partition. Image segmentation is one way to extract object information. It decomposes an image into a collection of regions, which correspond to objects if decomposition is ideal. Image segmentation has been successfully used in content-based image retrieval [10, 16, 66, 99, 120, 130]. Several

[1] As defined in [120], a graph image is an image containing mainly text, graph, and overlays; a photograph is a continuous-tone image.

region-based methods have been developed for image categorization as well. SIMPLIcity system [120] classifies images into *textured* or *non-textured* classes based upon how evenly a region scatters in an image. Mathematically, this is described by the goodness of match, which is measured by the χ^2 statistic, between the distribution of the region and a uniform distribution. Smith and Li [99] proposed a method for classifying images by spatial orderings of regions. Their system decomposes an image into regions with the attribute of interest of each region represented by a symbol that corresponds to an entry in a finite pattern library. Each region string is converted to a composite region template (CRT) descriptor matrix that enables classification using spatial information. Since a CRT matrix is determined solely by the ordering of symbols, the method is sensitive to object shifting and rotation. The work by Barnard and Forsyth [5] achieves some success in associating words to images based on regions. In this method, an image is modeled as a sequence of regions and a sequence of words generated by a hierarchical statistical model, which describes the occurrence and co-occurrence of region features and object names.

Chapter 3

Related Work in Machine Learning

In this chapter, we present a brief review of three machine learning techniques, Support Vector Machines (SVMs), additive fuzzy systems, and spectral graph clustering, which will be applied in the subsequent chapters.

3.1 Support Vector Machines

This section presents the basic concepts of the VC theory and SVMs. For gentle tutorials, we refer interested readers to Burges [9]. More exhaustive treatments can be found in the books by Vapnik [114, 115].

3.1.1 VC Theory

Let's consider a two-class classification problem of assigning label $y \in \{+1, -1\}$ to input feature vector $\mathbf{x} \in \mathbb{R}^n$. We are given a set of training samples $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l)\}$ that are drawn independently from some unknown cumulative probability distribution $P(\mathbf{x}, y)$. The learning task is formulated as finding a machine (a function $f : \mathbb{R}^n \rightarrow \{+1, -1\}$) that "best" approximates the mapping generating the training set. For any feature vector $\mathbf{x} \in \mathbb{R}^n$, $f(\mathbf{x}) \in \{+1, -1\}$ is the predicted class label for $\mathbf{x}$. In order to make learning feasible, we need to specify a function space, $\mathbb{H}$, from which a machine is chosen. $\mathbb{H}$ can be the set of hyperplanes in $\mathbb{R}^n$, polynomials of degree d, artificial neural networks with certain structure, or, in general, a set of parameterized functions.

One way to measure performance of a selected machine f is to look at how it behaves on the training set. This can be quantified by the empirical risk (or training error)

$$R_{emp}(f) = \frac{1}{l}\sum_{i=1}^{l} \mathrm{I}_{\{f(\mathbf{x}_i)\neq y_i\}}(\mathbf{x}_i, y_i)$$

where $\mathrm{I}_A(z)$ is an indicator function defined as $\mathrm{I}_A(z) = 1$ for all $z \in A$, and $\mathrm{I}_A(z) = 0$ for all $z \notin A$. Although the empirical risk can be minimized to zero if $\mathbb{H}$ and learning algorithm are properly chosen, the resulting f may not make correct classifications of unseen data. The ability of f to correctly classify data not in the training set is known as generalization. It is this property that we shall aim to optimize. Therefore, a better performance measure for f is

$$R_{P(\mathbf{x},y)}(f) = \int_{\mathbb{R}^n\times\{+1,-1\}} \mathrm{I}_{\{f(\mathbf{x})\neq y\}}(\mathbf{x}, y)dP(\mathbf{x}, y) \quad . \tag{3.1}$$

$R_{P(\mathbf{x},y)}(f)$ is called the expected risk (the probability of misclassifications made by f). Unfortunately, equation (3.1) is more an elegant way of writing the error probability than practical usefulness because $P(\mathbf{x}, y)$ is usually unknown.

However, there is a family of bounds on the expected risk, which demonstrates fundamental principles of building machines with good generalization. Here we present one result from the VC theory due to Vapnik and Chervonenkis [116]: given a set of l training samples and function space $\mathbb{H}$, with probability $1-\eta$, for any $f \in \mathbb{H}$ the expected

risk is bounded above by

$$R_{P(\mathbf{x},y)}(f) \leq R_{emp}(f) + \sqrt{\frac{h(1+\ln\frac{2l}{h}) - \ln\frac{\eta}{4}}{l}} \tag{3.2}$$

for any distribution $P(\mathbf{x}, y)$ on $\mathbb{R}^n \times \{+1, -1\}$. Here h is a non-negative integer called the Vapnik Chervonenkis (VC) dimension, or in short VC dimension. It is a measure of the capacity of a $\{+1, -1\}$-valued function space (in our case $\mathbb{H}$), and is defined as the size of the largest subset of domain points that can be labeled arbitrarily (or called shattered) by choosing functions only from $\mathbb{H}$. Note that the right hand side of (3.2) is distribution free. If we know h, we can derive an upper bound for $R_{P(\mathbf{x},y)}(f)$ that is usually not likely to compute. Moreover, given a training set of size l, (3.2) demonstrates a strategy to control expected risk by controlling two quantities: the empirical risk and the VC dimension. For a given function space, its VC dimension is fixed. Thus the lowest upper bound is achieved by selecting a machine (using some learning algorithm) that minimizes the empirical risk. The same procedure can be done for different function spaces with different VC dimensions. We then choose a machine that gives the lowest upper bound across all the given function spaces [1]. Next we will discuss an application of this idea: the SVM learning strategy.

3.1.2 Support Vector Machines

Let $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l)\} \subset \mathbb{R}^n \times \{+1, -1\}$ be a training set, and $\langle \cdot, \cdot \rangle$ be an inner product in $\mathbb{R}^n$ defined as $\langle \mathbf{x}, \mathbf{z} \rangle = \mathbf{x}^T\mathbf{z}$. If the classes are linearly separable, then

[1] This is the basic idea behind structural risk minimization.

there exists a hyperplane $\left\{\mathbf{x} \in \mathbb{R}^n : \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\right\}$ and the induced classification rule, $f : \mathbb{R}^n \to \{+1, -1\}$,

$$f(\mathbf{x}) = \text{sign}\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right) \tag{3.3}$$

such that the training samples are correctly classified by f, and

$$\min_{i=1,\cdots,l} |\langle \mathbf{w}, \mathbf{x_i} \rangle + b| = 1 \ .$$

Geometrically, this can be illustrated in Figure 3.1 where the hyperplane (a straight line in the figure) corresponding to $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ is the decision boundary, and the region $\{\mathbf{x} \in \mathbb{R}^n : |\langle \mathbf{w}, \mathbf{x} \rangle + b| \leq 1\}$ is bounded by the hyperplanes above and below the decision boundary. The distance between these two bounding hyperplanes is called the margin between the two classes on the training data under a separating hyperplane. It is defined as $\frac{2}{\langle \mathbf{w}, \mathbf{w} \rangle}$. Clearly, different $\mathbf{w}$'s give different margins. For generalization purpose, as we will see shortly, it is desirable to find the maximal separating hyperplane: the hyperplane that creates the biggest margin (the decision boundary in Figure 3.1 is in fact the maximal separating hyperplane). This leads to the following convex optimization problem:

$$\begin{aligned} &\min_{\mathbf{w},b} \frac{1}{2} \langle \mathbf{w}, \mathbf{w} \rangle \\ \text{subject to} \quad &y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) \geq 1, \ \forall i \quad . \end{aligned} \tag{3.4}$$

Minimizing $\frac{1}{2} \langle \mathbf{w}, \mathbf{w} \rangle$ is equivalent to maximizing the margin. The constraints $y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) \geq 1 \ \forall i$ imply correct separation.

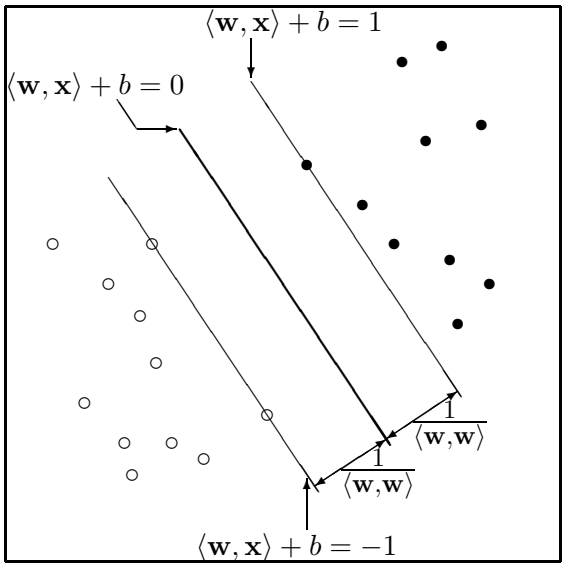


Fig. 3.1. Optimal separating hyperplane $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ with maximal margin $\frac{2}{\langle \mathbf{w}, \mathbf{x} \rangle}$.

In practice, however, a separating hyperplane does not exist if the two classes are linearly inseparable. One way to deal with this is to modify the constraints to allow for the possibility of misclassifications. Define the nonnegative slack variables $\xi = [\xi_1, \cdots, \xi_l]^T$. The constraints in (3.4) is modified as

$$y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) \geq 1 - \xi_i, \ \forall i \quad .$$

The value ξ_i is the distance by which $\mathbf{x}_i$ is on the wrong side of its margin. Misclassifications occur when $\xi_i > 1$, so bounding $\sum_{i=1}^{l} \xi_i$ limits the total number of training errors. Therefore, the optimal separating hyperplane is found by solving the following

quadratic program:

$$\min_{\mathbf{w},b} \frac{1}{2} \langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{i=1}^{l} \xi_i \tag{3.5}$$
$$\text{subject to} \qquad y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) \geq 1 - \xi_i, \ \xi_i \geq 0, \ \forall i$$

where $C > 0$ is some constant.

How does minimizing (3.5) relate to our ultimate goal of optimizing the generalization? To answer this question, we need to introduce a theorem [114] about the VC dimension of a class of functions $\mathbb{H} = \{f(\mathbf{x}) : \langle \mathbf{w}, \mathbf{w} \rangle \leq A\}$ where f is defined in (3.3). One can show that for a given set of training samples contained in a sphere of radius R, the VC dimension h of the function space $\mathbb{H}$ is bounded above by

$$h \leq \min \left(R^2 A^2, n \right) + 1 \quad .$$

Thus, minimizing the quadratic term $\frac{1}{2} \langle \mathbf{w}, \mathbf{w} \rangle$ amounts to minimizing the VC dimension of $\mathbb{H}$ from which the classification rule is chosen, therefore minimizing the second term of the bound (3.2). On the other hand, $\sum_{i=1}^{l} \xi_i$ is an upper bound on the number of misclassifications on the training set, thus controls the empirical risk term in (3.2). For an adequate positive constant C, minimizing (3.5) can indeed decrease the upper bound on the expected risk.

Applying the Karush-Kuhn-Tucker conditions, one can show that any $\mathbf{w}$, which minimizes (3.5), can be written as a linear combination of the training samples

$$\mathbf{w} = \sum_{i=1}^{l} y_i \alpha_i \mathbf{x}_i \quad . \tag{3.6}$$

The above expansion is called the dual representation of $\mathbf{w}$, in which the number of unknown coefficients, which are Lagrange multipliers, equals the number of training samples. A coefficient α_j is nonzero only if $y_j \left(\left\langle \mathbf{w}, \mathbf{x}_j \right\rangle + b \right) = 1 - \xi_i$. These $\mathbf{x}_j$'s are called support vectors. The index set of support vectors is denoted by $\mathcal{S}$. Substituting (3.6) into (3.3), we obtain the optimal decision rule

$$f(\mathbf{x}) = \text{sign} \left(\sum_{i \in \mathcal{S}} y_i \alpha_i \left\langle \mathbf{x}, \mathbf{x}_i \right\rangle + b \right) \tag{3.7}$$

where α_i's can be found by solving the Wolfe dual problem [124] of (3.5) (the dual problem is a simpler convex quadratic programming problem than the primal)

$$\max_{\alpha} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle \tag{3.8}$$

$$\text{subject to} \qquad C \geq \alpha_i \geq 0, \ \forall i, \ \sum_{i=1}^{l} \alpha_i y_i = 0.$$

Given α, b can be determined by solving $\sum_{i \in \mathcal{S}} y_i \alpha_i \left\langle \mathbf{x}_j, \mathbf{x}_i \right\rangle + b = y_j$ for any (or all) $\mathbf{x}_j$, $0 < \alpha_j < C$.

The SVMs described so far finds linear boundaries in the input feature space $\mathbb{R}^n$. More complex decision surfaces can be generated by employing a nonlinear mapping

$\Phi : \mathbb{R}^n \to \mathbb{F}$ to map the data into a new feature space $\mathbb{F}$, usually with dimension higher than n, and solving the same optimization problem in $\mathbb{F}$, i.e., find the maximal separating hyperplane in $\mathbb{F}$. Note that in (3.7) and (3.8) $\mathbf{x}_i$ never appears isolated but always in the form of an inner product $\left\langle \mathbf{x}, \mathbf{x}_j \right\rangle$ (or $\left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle$). This implies that there is no need to evaluate the nonlinear mapping Φ as long as we know the inner product in $\mathbb{F}$ for any given $\mathbf{x}, \mathbf{z} \in \mathbb{R}^n$. For computational purposes, instead of defining $\Phi : \mathbb{R}^n \to \mathbb{F}$ explicitly, a function $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is introduced to directly define an inner product in $\mathbb{F}$, i.e., $K(\mathbf{x}, \mathbf{z}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle_{\mathbb{F}}$ where $\langle \cdot, \cdot \rangle_{\mathbb{F}}$ is an inner product in $\mathbb{F}$, and Φ is a nonlinear mapping induced by K. Such a function K is also called a Mercer kernel [24, 114, 115], which will be explored further in the next section. Substituting $K(\mathbf{x}_i, \mathbf{x}_j)$ for $\left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle$ in (3.8) produces a new optimization problem

$$\max_{\alpha} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{3.9}$$

$$\text{subject to} \qquad C \geq \alpha_i \geq 0, \ \forall i, \ \sum_{i=1}^{l} \alpha_i y_i = 0.$$

Solving (3.9) for α gives a decision rule of the form

$$f(\mathbf{x}) = \text{sign} \left(\sum_{i \in \mathcal{S}} y_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b \right) \quad , \tag{3.10}$$

whose decision boundary is a hyperplane in $\mathbb{F}$, and translates to nonlinear boundaries in the original space. Several techniques for solving quadratic programming problems arising in SVMs are described in [55, 56, 82].

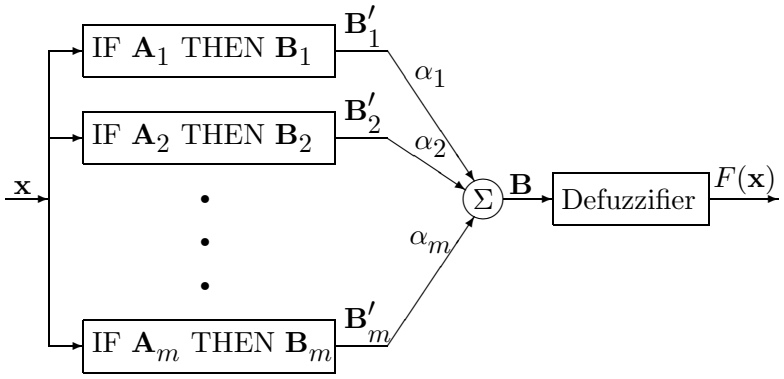


Fig. 3.2. Architecture of an additive fuzzy system.

3.2 Additive Fuzzy Systems

Since the publication of L.A. Zadeh's seminal paper on fuzzy sets [129], fuzzy set theory and its descendant, fuzzy logic, have evolved into powerful tools for managing uncertainties inherent in complex systems. In the recent twenty years, fuzzy methodology has been successfully applied to a variety of areas including control and system identification [58, 62, 107, 122, 134], signal and image processing [78, 91, 103], pattern classification [1, 44, 50, 57], and information retrieval [14, 75].

An additive fuzzy system F stores m fuzzy rules of the form "IF $X = \mathbf{A}_j$ THEN $Y = \mathbf{B}_j$" and computes the output $F(\mathbf{x})$ by defuzzifying the summed and partially fired THEN-part fuzzy sets [60]. In general, an additive fuzzy system acts as a multiple-input multiple-output (MIMO) mapping, $F : \mathbb{R}^n \to \mathbb{R}^p$. In this section, however, we focus on multiple-input single-output (MISO) models. The results derived here still apply to the MIMO models by combining several MISO models provided that no coupling exists among outputs.

Figure 3.2 shows the "parallel fire-and-sum" structure of an additive fuzzy system [60]. Each input $\mathbf{x} \in \mathbb{R}^n$ activates all m IF-part fuzzy sets to degrees $a_j(\mathbf{x}) \in [0,1]$, $j = 1, \cdots, m$, which in turn scale the THEN-part fuzzy sets $\mathbf{B}_j$ to produce $\mathbf{B}'_j$. The output set $\mathbf{B}$ is computed as a weighted sum of $\mathbf{B}'_j$, and is defined by a set function $b : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^+$ as

$$b(\mathbf{x}, y) = \sum_{j=1}^{m} \alpha_j a_j(\mathbf{x}) b_j(y) \tag{3.11}$$

where $b_j : \mathbb{R} \to [0,1]$ is the membership function for $\mathbf{B}_j$, $\alpha_j \geq 0$ is the weight for the jth fuzzy rule. The system defuzzifies $\mathbf{B}$ to give the output $y = F(\mathbf{x})$. A fuzzy rule is called active if its weight is nonzero.

Although an additive fuzzy system allows us to pick arbitrary IF-part fuzzy sets, factorable fuzzy sets are most commonly employed in practice [60, 74]. An n dimensional fuzzy set [2] is factorable if and only if it can be written as the Cartesian product of n scalar fuzzy sets. For example, if $\mathbf{A}_j$ is factorable with membership function $a_j : \mathbb{R}^n \to [0,1]$, then it can be equivalently written as $\mathbf{A}_j = \mathbf{A}_j^1 \times \mathbf{A}_j^2 \times \cdots \times \mathbf{A}_j^n$ with membership function

$$a_j(\mathbf{x}) = a_j^1(x_1) \otimes a_j^2(x_2) \otimes \cdots \otimes a_j^n(x_n) \tag{3.12}$$

where $\mathbf{x} = [x_1, x_2, \cdots, x_n]^T \in \mathbb{R}^n$, $\mathbf{A}_j^k$ is a scalar fuzzy set with membership function $a_j^k : \mathbb{R} \to [0,1]$, $k = 1, \cdots, n$, $\times$ denotes the Cartesian product, and $\otimes$ represents the fuzzy conjunction operator. As a result, we interpret the fuzzy rule "IF $\mathbf{A}_j$ THEN $\mathbf{B}_j$"

[2] An n dimensional fuzzy set is a fuzzy set in $\mathbb{R}^n$ with membership function $a : \mathbb{R}^n \to [0,1]$.

as

$$\text{IF } \mathbf{A}_j^1 \text{ AND } \mathbf{A}_j^2 \text{ AND } \cdots \text{ AND } \mathbf{A}_j^n \text{ THEN } \mathbf{B}_j \quad . \tag{3.13}$$

The fuzzy conjunction (AND) operator can be chosen freely from the set of t-norms [62], though product and min operators are often employed.

Intuitively, the output set $\mathbf{B}$ describes the output distribution for a given input. Nevertheless, in many applications a crisp output value is required. For example, the output of a fuzzy classifier should be the class label corresponding to a given input, while the prediction made by a fuzzy function approximator is usually a real number. The mapping from $\mathbf{B}$ to some real number is realized by a defuzzifier. Several commonly used defuzzification strategies may be described as the max criterion (MC), the mean of maximum (MOM), and the center of area (COA) [62]. For a given input $\mathbf{x}$, the MC finds the global maximizer of $b(\mathbf{x}, y)$, the MOM computes the mean value of all local maximizers of $b(\mathbf{x}, y)$, and COA defines the output as

$$\frac{\int_{-\infty}^{\infty} y b(\mathbf{x}, y) dy}{\int_{-\infty}^{\infty} b(\mathbf{x}, y) dy} \ .$$

Consider an additive fuzzy system with m fuzzy rules of the form

$$\text{Rule } j: \quad \text{IF } \mathbf{A}_j^1 \text{ AND } \mathbf{A}_j^2 \text{ AND } \cdots \text{ AND } \mathbf{A}_j^n \text{ THEN } b_j \tag{3.14}$$

where $\mathbf{A}_j^k$ is a fuzzy set with membership function $a_j^k : \mathbb{R} \to [0, 1]$, $j = 1, \cdots, m$, $k = 1, \cdots, n$, $b_j \in \mathbb{R}$. If we choose product as the fuzzy conjunction operator and COA defuzzification, then the model becomes a special form of the Takagi-Sugeno (TS) fuzzy

model [107], and the input output mapping, $F : \mathbb{R}^n \to \mathbb{R}$, of the model is defined as

$$F(\mathbf{x}) = \frac{\sum_{j=1}^{m} b_j \prod_{k=1}^{n} a_j^k(x_k)}{\sum_{j=1}^{m} \prod_{k=1}^{n} a_j^k(x_k)} \tag{3.15}$$

where $\mathbf{x} = [x_1, \cdots, x_n]^T \in \mathbb{R}^n$ is the input. Note that (3.15) is not well-defined on $\mathbb{R}^n$ if $\sum_{j=1}^{m} \prod_{k=1}^{n} a_j^k(x_k) = 0$ for some $\mathbf{x} \in \mathbb{R}^n$, which could happen if the input space is not wholly covered by fuzzy rule "patches." However, there are several straightforward solutions for this problem. For example, we can force the output to some constant when $\sum_{j=1}^{m} \prod_{k=1}^{n} a_j^k(x_k) = 0$, or add a fuzzy rule so that the denominator $\sum_{j=1}^{m} \prod_{k=1}^{n} a_j^k(x_k) > 0$ for all $\mathbf{x} \in \mathbb{R}^n$. Here we take the second approach for analytical simplicity. The following rule is added:

$$\text{Rule } 0: \quad \text{IF } \mathbf{A}_0^1 \text{ AND } \mathbf{A}_0^2 \text{ AND } \cdots \text{ AND } \mathbf{A}_0^n \text{ THEN } b_0 \tag{3.16}$$

where $b_0 \in \mathbb{R}$, the membership functions $a_0^k(x_k) \equiv 1$ for $k = 1, \cdots, n$ and any $x_k \in \mathbb{R}$. Consequently, the input output mapping becomes

$$F(\mathbf{x}) = \frac{b_0 + \sum_{j=1}^{m} b_j \prod_{k=1}^{n} a_j^k(x_k)}{1 + \sum_{j=1}^{m} \prod_{k=1}^{n} a_j^k(x_k)} \quad . \tag{3.17}$$

3.3 Spectral Graph Clustering

Data representation is typically the first step to solve any clustering problem. In the field of computer vision, two types of representations are widely used. One is called *geometric representation*, in which data items are mapped to some real normed

vector space. The other, referred to as *graph representation*, emphasizes the pairwise relationship, but is usually short of geometric interpretation.

Under graph representation, a collection of n data samples can be represented by a weighted undirected graph $G = (\mathbf{V}, \mathbf{E})$: the nodes $\mathbf{V} = \{1, 2, \ldots, n\}$ represent data samples, the edges $\mathbf{E} = \{(i, j) : i, j \in \mathbf{V}\}$ are formed between every pair of nodes, and the nonnegative weight w_{ij} of an edge (i, j), indicating the similarity between two nodes, is a function of the distance (or similarity) between nodes i and j. The weights can be organized into a matrix $\mathbf{W}$, named *affinity matrix*, with the ij-th entry denoted by w_{ij}.

Under a graph representation, clustering can be naturally formulated as a graph partitioning problem. Among many graph-theoretic algorithms, spectral graph partitioning methods [22, 80, 90, 96, 123] have been successfully applied to many areas in computer vision including motion analysis [22], image segmentation [96, 123], and object recognition [90]. In this paper, we use one of the techniques, the normalized cut (Ncut) method [96], for image clustering. Compared with many other spectral graph partitioning methods, such as average cut and average association, the Ncut method is empirically shown to be relatively robust for image segmentation applications [96, 123]. Next, we present a brief review of the Ncut method based on Shi and Malik's work [96]. More exhaustive treatments can be found in [96] and [123].

Roughly speaking, a graph partitioning method attempts to organize nodes into groups so that the within-group similarity is high, and/or the between-groups similarity is low. Given a graph $G = (\mathbf{V}, \mathbf{E})$ with affinity matrix $\mathbf{W}$, a simple way to quantify the cost for partitioning nodes into two disjoint sets $\mathbf{A}$ and $\mathbf{B}$ ($\mathbf{A} \cap \mathbf{B} = \emptyset$ and $\mathbf{A} \cup \mathbf{B} = \mathbf{V}$) is the total weights of the edges that connect the two sets. In the terminology of the

graph theory, it is called the *cut*:

$$cut(\mathbf{A}, \mathbf{B}) = \sum_{i \in \mathbf{A}, j \in \mathbf{B}} w_{ij} \ , \tag{3.18}$$

which can also be viewed as a measure of the between-groups similarity.

Finding a bipartition of the graph that minimizes this cut value is known as the *minimum cut* problem. There exist efficient algorithms for solving this problem. However the minimum cut criterion favors grouping small sets of isolated nodes in the graph [96] because the cut defined in (3.18) does not contain any within-group information. In other words, the minimum cut usually yields over-clustered results when it is recursively applied. This motivates several modified graph partition criteria including the Ncut:

$$Ncut(\mathbf{A}, \mathbf{B}) = \frac{cut(\mathbf{A}, \mathbf{B})}{assoc(\mathbf{A}, \mathbf{V})} + \frac{cut(\mathbf{A}, \mathbf{B})}{assoc(\mathbf{B}, \mathbf{V})}$$

where $assoc(\mathbf{A}, \mathbf{V}) = \sum_{i \in \mathbf{A}, j \in \mathbf{V}} w_{ij}$ is the total weights of the edges that connect nodes in $\mathbf{A}$ to all nodes in the graph and $assoc(\mathbf{B}, \mathbf{V})$ is defined similarly. Note that the $Ncut$ value is always within the interval $[0, 1]$. An unbalanced cut would make the Ncut value very close to 1 since $assoc(\mathbf{A}, \mathbf{V}) = cut(\mathbf{A}, \mathbf{B}) + assoc(\mathbf{A}, \mathbf{A})$ and $assoc(\mathbf{B}, \mathbf{V}) = cut(\mathbf{A}, \mathbf{B}) + assoc(\mathbf{B}, \mathbf{B})$.

Shi and Malik [96] have shown that finding a bipartition with minimum Ncut value can be formulated as the following discrete optimization problem:

$$\mathbf{y} = \arg\min_{\mathbf{y}} \frac{\mathbf{y}^T(\mathbf{D} - \mathbf{W})\mathbf{y}}{\mathbf{y}^T\mathbf{D}\mathbf{y}} \tag{3.19}$$

with the constraints: 1) $\mathbf{y} \in \{1, -b\}^n$, $b > 0$; and 2) $\mathbf{y}^T\mathbf{D1} = 0$. Here $\mathbf{W}$ is an $n \times n$ affinity matrix, $\mathbf{D} = diag[s_1, s_2, \cdots, s_n]$ is a diagonal matrix with $s_i = \sum_{j=1,\cdots,n} w_{ij}$, and $\mathbf{1}$ is a vector of all ones. The partition is decided by $\mathbf{y}$: if the i-th element of $\mathbf{y}$ is greater than zero then node i is in $\mathbf{A}$, otherwise in $\mathbf{B}$. Unfortunately, solving the above discrete optimization problem is NP-complete [96]. However, Shi and Malik show that if the first constraint on $\mathbf{y}$ is relaxed, i.e., $\mathbf{y}$ can take real values, then the continuous version of (3.19) can be minimized by solving the generalized eigenvalue system:

$$(\mathbf{D} - \mathbf{W})\mathbf{y} = \lambda\mathbf{D}\mathbf{y} \; . \tag{3.20}$$

And the solution is the generalized eigenvector corresponding to the second smallest generalized eigenvalue (or in short the second smallest generalized eigenvector). Even though there is no guarantee that this continuous approximation will be close to the correct discrete solution, abundant experimental evidence demonstrates that the second smallest generalized eigenvector does carry useful grouping information [96, 123], and therefore is used by the Ncut method to bipartition the graph.

Unlike the ideal case, in which the signs of the values in the eigenvector can decide the partition since the eigenvector can only take on two discrete values, the second smallest generalized eigenvector of (3.20) usually takes on continuous values. Several ways have been proposed in [96] to choose a splitting point: 1) keep 0 as the splitting point; 2) use the median value of the second smallest generalized eigenvector as the splitting point; 3) check l possible splitting points that are evenly spaced between the

minimum and maximum values of the second smallest generalized eigenvector, and pick the one with the minimum Ncut value. The last approach is employed in this work.

Next we would like to point out some implementation details: finding the second smallest generalized eigenvector is equivalent to computing the largest eigenvector of a transformed affinity matrix. It is not difficult to verify that the eigenvalues of

$$\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-\frac{1}{2}}$$

are identical to the generalized eigenvalues of (3.20). Moreover, if $\mathbf{y}$ is a generalized eigenvector of (3.20) then

$$\mathbf{z} = \mathbf{D}^{\frac{1}{2}}\mathbf{y} \tag{3.21}$$

is an eigenvector of $\mathbf{L}$ for the same eigenvalue (or generalized eigenvalue). Therefore one can alternatively compute the second smallest eigenvector of $\mathbf{L}$ and transform it to the desired generalized eigenvector using (3.21). The matrix $\mathbf{L}$, which is a normalized *Laplacian matrix* ($\mathbf{D} - \mathbf{W}$ is called the Laplacian matrix [83]), has the following properties: 1) it is positive semidefinite with all the eigenvalues in the interval $[0, 2]$; 2) 0 and $\mathbf{z}_0 = \mathbf{D}^{\frac{1}{2}}\mathbf{1}$ are the smallest eigenvalue and eigenvector, respectively. From these properties, it is clear that if λ and $\mathbf{z}$ are an eigenvalue and eigenvector of $\mathbf{L}$, respectively, then $2 - \lambda$ and $\mathbf{z}$ are eigenvalue and eigenvector of $2\mathbf{I} - \mathbf{L}$, respectively, where $\mathbf{I}$ is an identity matrix. Moreover, 2 is the largest eigenvalue of $2\mathbf{I} - \mathbf{L}$ whose second largest eigenvalue corresponds to the second smallest eigenvalue of $\mathbf{L}$. Subtracting from $2\mathbf{I} - \mathbf{L}$ a rank-one matrix defined by its largest eigenvalue 2 and unit length eigenvector $\frac{\mathbf{z}_0}{\|\mathbf{z}_0\|}$

gives

$$\mathbf{L}^* = 2\mathbf{I} - \mathbf{L} - \frac{2}{\|\mathbf{z}_0\|^2}\mathbf{z}_0\mathbf{z}_0^T \ .$$

It is straightforward to check that the largest eigenvector of $\mathbf{L}^*$ is the second smallest eigenvector of $\mathbf{L}$. The cost to compute $\mathbf{L}^*$ is very low since $\mathbf{D}$ is a diagonal matrix with positive diagonal entries. Therefore, one can apply an eigensolver, such as the Lanczos method (Ch.9, [39]), to $\mathbf{L}^*$ directly.

Chapter 4

Support Vector Learning for Fuzzy Rule-Based Classification Systems

SVM method described in Chapter 3.1 represents one of the most important directions both in theory and application of machine learning. While fuzzy classifier was regarded as a method that "are cumbersome to use in high dimensions or on complex problems or in problems with dozens or hundreds of features (pp. 194, [29])." In this chapter, we investigate the connections between these two seemingly unrelated areas. The result of this chapter will be used in Chapter 7 to design an image categorization algorithm.

4.1 Overview

In general, building a fuzzy system consists of three basic steps [125]: structure identification (variable selection, partitioning input and output spaces, specifying the number of fuzzy rules, and choosing a parametric/nonparametric form of membership functions), parameter estimation (obtaining unknown parameters in fuzzy rules via optimizing a given criterion), and model validation (performance evaluation and model simplification).

Deciding the number of input variables is referred to as the problem of variable selection, i.e., selecting input variables that are most predictive of a given outcome. Given a set of input and output variables, a fuzzy partition associates fuzzy sets with

each variable. There are roughly two ways of doing it: data independent partition and data dependent partition. The former approach partitions the input space in a predetermined fashion. One of the commonly used strategies is to assign fixed number of linguistic labels to each input variable. The partition of the output space then follows from supervised learning. Although this scheme is simple to implement, it has two severe drawbacks:

- The performance of the resulting system may be very bad if the input space partition is quite distinct from the distribution of data. Optimizing output space partition alone is not sufficient.
- It suffers from the curse of dimensionality. If each input variable is allocated m fuzzy sets, a fuzzy system with n inputs and one output needs on the order of m^n rules.

Various data dependent partition methods have been proposed to alleviate these drawbacks. They are basically based on data clustering techniques [26, 94, 109].

Although a fuzzy partition can generate fuzzy rules, results are usually very coarse with many parameters needing to be learned and tuned. Various optimization techniques are proposed to solve this problem. Genetic algorithms [21] and artificial neural networks [54] are two of the most popular and effective approaches.

After going through the long journey of structure identification and parameter estimation, can we infer that we get a good fuzzy model? Conclusions could not be drawn without answering the following two questions:

- How capable can a fuzzy model be?

- How well can the model, built on finite amount of data, capture the concept underlying the data?

The first question could be answered from the perspective of function approximation. Several types of fuzzy models are proven to be "universal approximators" [85, 127]. The second question is about the generalization performance, which is closely related to several well-known problems in the statistics and machine learning literature, such as the structural risk minimization (SRM) [113], the bias variance dilemma [35], and the overfitting phenomena [6]. Loosely speaking, a model, built on finite amount of training data, generalizes the best if the right tradeoff is found between the training accuracy and the "capacity" of the model set from which the model is chosen. On one hand, a low "capacity" model set may not contain any model that fits the training data well. On the other hand, too much freedom may eventually generate a model behaving like a refined look-up-table: perfect for the training data but (maybe) poor on generalization.

Researchers in the fuzzy systems community attempt to tackle this problem with roughly two approaches:(1) use the idea of cross-validation to select a model that has the best ability to generalize [102]; (2) focus on model reduction, which is usually achieved by rule base reduction [126], to simplify the model. In statistical learning literature, the Vapnik-Chervonenkis (VC) theory [115] provides a general measure of model set complexity, and gives associated bounds on generalization. However, no efforts have been made to apply the VC theory and the related technique, SVM, to construct fuzzy systems. The work presented in this chapter tries to bridge this gap.

4.2 Additive Fuzzy Rule-Based Classification Systems

A classifier associates class labels with input features, i.e., it is essentially a mapping from the input space to the set of class labels. In this thesis, we are interested in binary fuzzy classifiers defined as follows.

Definition 4.1. *(Binary Fuzzy Classifier) Consider a fuzzy system with* $m+1$ *fuzzy rules where* Rule 0 *is given by (3.16),* $j = 1, \cdots, m$, Rule j *has the form of (3.14). If the system uses product for fuzzy conjunction, addition for rule aggregation, and COA defuzzification, then the system induces a binary fuzzy classifier,* f*, with decision rule,*

$$f(\mathbf{x}) = \text{sign}\left(F(\mathbf{x}) + t\right) \tag{4.1}$$

where $F(\mathbf{x})$ *is defined in (3.17),* $t \in \mathbb{R}$ *is a threshold.*

The following corollary states that, without loss of generality, we can assume $t = 0$.

Corollary 4.2. *For any binary fuzzy classifier given by Definition 4.1 with nonzero threshold* t*, there exists a binary fuzzy classifier that has the same decision rule but zero threshold.*

Proof: Suppose we are given a binary fuzzy classifier, f, with $t \neq 0$. From (3.17) and (4.1), we have

$$f(\mathbf{x}) = \text{sign}\left(\frac{(b_0 + t) + \sum_{j=1}^{m}(b_j + t)\prod_{k=1}^{n} a_j^k(x_k)}{1 + \sum_{j=1}^{m}\prod_{k=1}^{n} a_j^k(x_k)}\right) \quad ,$$

which is identical to the decision rule of a binary fuzzy classifier with $b_j + t$ as the THEN-part of jth fuzzy rule ($j = 0, \cdots, m$) and zero threshold. □

The membership functions for a binary fuzzy classifier defined above could be any function from $\mathbb{R}$ to $[0, 1]$. However, too much flexibility on the model could make effective learning (or training) unfeasible. Therefore we narrow our interests to a class of membership functions, which are generated from location transformation of reference functions [28], and the classifiers defined on them.

Definition 4.3. *(Reference Function [1], [28]) A function $\mu : \mathbb{R} \to [0, 1]$ is a reference function if and only if $\mu(x) = \mu(-x)$ and $\mu(0) = 1$.*

Definition 4.4. *(Standard Binary Fuzzy Classifier) A binary fuzzy classifier given by Definition 4.1 is a standard binary fuzzy classifier if for the kth input, $k \in \{1, \cdots, n\}$, the membership functions, $a_j^k : \mathbb{R} \to [0, 1]$, $j = 1, \cdots, m$, are generated from a reference function a^k through location transformation, i.e., $a_j^k(x_k) = a^k(x_k - z_j^k)$ for some location parameter $z_j^k \in \mathbb{R}$.*

Definition 4.5. *(Translation Invariant Kernel) A kernel $K(\mathbf{x}, \mathbf{z})$ is translation invariant if $K(\mathbf{x}, \mathbf{z}) = K(\mathbf{x} - \mathbf{z})$, i.e., it depends only on $\mathbf{x} - \mathbf{z}$, but not on $\mathbf{x}$ and $\mathbf{z}$ themselves.*

Corollary 4.6. *The decision rule of a standard binary fuzzy classifier given by Definition 4.4 can be written as*

$$f(\mathbf{x}) = \text{sign}\left(\sum_{j=1}^{m} b_j K(\mathbf{x}, \mathbf{z}_j) + b_0\right) \tag{4.2}$$

[1] Note that the original definition in [28] has an extra condition: *μ is nonincreasing on $[0, \infty)$*. But this condition is not needed in deriving our results, and therefore, is omitted.

where $\mathbf{x} = [x_1, x_2, \cdots, x_n]^T \in \mathbb{R}^n$, $\mathbf{z}_j = [z_j^1, z_j^2, \cdots, z_j^n]^T \in \mathbb{R}^n$ *contains the location parameters of* a_j^k, $k = 1, \cdots, n$, $K : \mathbb{R}^n \times \mathbb{R}^n \to [0, 1]$ *is a translation invariant kernel defined as*

$$K(\mathbf{x}, \mathbf{z}_j) = \prod_{k=1}^{n} a^k(x_k - z_j^k) \quad . \tag{4.3}$$

Proof: From (3.17), (4.1), and Corollary 4.2, the decision rule of a binary fuzzy classifier is

$$f(\mathbf{x}) = \text{sign}\left(\frac{b_0 + \sum_{j=1}^{m} b_j \prod_{k=1}^{n} a_j^k(x_k)}{1 + \sum_{j=1}^{m} \prod_{k=1}^{n} a_j^k(x_k)} \right) \quad .$$

Since $1 + \sum_{j=1}^{m} \prod_{k=1}^{n} a_j^k(x_k) > 0$, we have

$$f(\mathbf{x}) = \text{sign}\left(b_0 + \sum_{j=1}^{m} b_j \prod_{k=1}^{n} a_j^k(x_k) \right) \quad . \tag{4.4}$$

From the definition of a standard binary fuzzy classifier, $a_j^k(x_k) = a^k(x_k - z_j^k)$, $k = 1, \cdots, n$, $j = 1, \cdots, m$. Substituting these results into (4.4) completes the proof. □

4.3 Positive Definite Fuzzy Classifiers

One particular kind of kernel, the Mercer kernel, has received considerable attention in the machine learning literature [24, 36, 115] because it is an efficient way of extending linear learning machines to nonlinear ones. Is the kernel defined by (4.3) a Mercer kernel? Before answering this question, we first quote a theorem.

Theorem 4.7. *(Mercer's Theorem [24, 72]) Let* $\mathbb{X}$ *be a compact subset of* $\mathbb{R}^n$. *Suppose* K *is a continuous symmetric function such that the integral operator* $T_K : L_2(\mathbb{X}) \to$

$L_2(\mathbb{X})$,

$$(T_K f)(\cdot) = \int_{\mathbb{X}} K(\cdot, \mathbf{x}) f(\mathbf{x}) d\mathbf{x}$$

is positive, that is

$$\int_{\mathbb{X}\times\mathbb{X}} K(\mathbf{x}, \mathbf{z}) f(\mathbf{x}) f(\mathbf{z}) d\mathbf{x} d\mathbf{z} \geq 0 \tag{4.5}$$

for all $f \in L_2(\mathbb{X})$. *Let* $\phi_i \in L_2(\mathbb{X})$, $i = 1,\ 2,\ \cdots$, *denote the eigenfunctions of the operator* T_K, *where each* ϕ_i *is normalized in such a way that* $\|\phi_i\|_{L_2} = 1$; *and let* λ_i, $i = 1,\ 2,\ \cdots$, *denote the corresponding eigenvalues. Then we can expand* $K(\mathbf{x}, \mathbf{z})$ *in a uniformly convergent series on* $\mathbb{X} \times \mathbb{X}$,

$$K(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{z}) \quad . \tag{4.6}$$

The positivity condition (4.5) is also called the Mercer condition. A kernel satisfying the Mercer condition is called a Mercer kernel. An equivalent form of the Mercer condition, which proves most useful in constructing Mercer kernels, is given by the following lemma [24].

Lemma 4.8. *(Positivity Condition for Mercer Kernels [24]) For a kernel* $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, *the Mercer condition (4.5) holds if and only if the matrix* $[K(\mathbf{x}_i, \mathbf{x}_j)] \in \mathbb{R}^{n\times n}$ *is positive semi-definite for all choices of points* $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\} \subset \mathbb{X}$ *and all* $n = 1, 2, \cdots\cdots$.

For most nontrivial kernels, directly checking the Mercer conditions in (4.5) or Lemma 4.8 is not an easy task. Nevertheless, for the class of translation invariant kernels, to which the kernels defined by (4.3) belong, there is an equivalent yet practically more powerful criterion based on the spectral property of the kernel [100].

Lemma 4.9. *(Mercer Conditions for Translation Invariant Kernels, Smola et al. [100]) A translation invariant kernel $K(\mathbf{x}, \mathbf{z}) = K(\mathbf{x} - \mathbf{z})$ is a Mercer kernel if and only if the Fourier transform*

$$\mathcal{F}[K](\omega) = \frac{1}{(2\pi)^{\frac{n}{2}}} \int_{\mathbb{R}^n} K(\mathbf{x}) e^{-i\langle \omega, \mathbf{x} \rangle} d\mathbf{x}$$

is nonnegative for all $\omega \in \mathbb{R}^n$.

Kernels defined by (4.3) do not, in general, have nonnegative Fourier transforms. However, if we assume that the reference functions are positive definite functions, which are defined by the following definition, then we do get a Mercer kernel (given in Theorem 4.12).

Definition 4.10. *(Positive Definite Function [47]) A function $f : \mathbb{R} \to \mathbb{R}$ is said to be a positive definite function if the matrix $[f(x_i - x_j)] \in \mathbb{R}^{n \times n}$ is positive semi-definite for all choices of points $\{x_1, \cdots, x_n\} \subset \mathbb{R}$ and all $n = 1, 2, \cdots\cdots$.*

Corollary 4.11. *A function $f : \mathbb{R} \to \mathbb{R}$ is positive definite if and only if the Fourier transform*

$$\mathcal{F}[f](\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx$$

is nonnegative all $\omega \in \mathbb{R}$.

Proof: Given any function $f : \mathbb{R} \to \mathbb{R}$, we can define a translation invariant kernel $K : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ as

$$K(x, z) = f(x - z) \quad .$$

From Lemma 4.9, K is a Mercer kernel if and only if the Fourier transform of f is nonnegative. Thus from Lemma 4.8 and Definition 4.10, we conclude that f is a positive definite function if and only if its Fourier transform is nonnegative. □

Theorem 4.12. *(Positive Definite Fuzzy Classifier, PDFC) A standard binary classifier given by Definition 4.4 is called a positive definite fuzzy classifier (PDFC) if the reference functions,* $a^k : \mathbb{R} \to [0,1],\ k = 1, \cdots, n$, *are positive definite functions. The translation invariant kernel (4.3) is then a Mercer kernel.*

Proof: From Lemma 4.9, it suffices to show that the translation invariant kernel defined by (4.3) has nonnegative Fourier transform. Rewrite (4.3) as

$$K(\mathbf{x}, \mathbf{z}) = K(\mathbf{u}) = \prod_{k=1}^{n} a^k(u_k)$$

where $\mathbf{x} = [x_1, \cdots, x_n]^T$, $\mathbf{z} = [z_1, \cdots, z_n]^T \in \mathbb{R}^n$, $\mathbf{u} = [u_1, \cdots, u_n]^T = \mathbf{x} - \mathbf{z}$. Then

$$\begin{aligned}
\mathcal{F}[K](\omega) &= \frac{1}{(2\pi)^{\frac{n}{2}}} \int_{\mathbb{R}^n} e^{-i\langle \omega, \mathbf{u} \rangle} \prod_{k=1}^{n} a^k(u_k) d\mathbf{u} \\
&= \frac{1}{(2\pi)^{\frac{n}{2}}} \int_{\mathbb{R}^n} \prod_{k=1}^{n} a^k(u_k) e^{-i\omega_k u_k} d\mathbf{u} \\
&= \prod_{k=1}^{n} \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} a^k(u_k) e^{-i\omega_k u_k} du_k
\end{aligned}$$

which is nonnegative since $a^k, k = 1, \cdots, n$, are positive definite functions (Corollary 4.11). □

4.4 Positive Definite Fuzzy Classifiers and Mercer Features

Recall the expansion (4.6) given by the Mercer Theorem. Let $\mathbb{F}$ be an l_2 space. If we define a nonlinear mapping $\Phi : \mathbb{X} \to \mathbb{F}$ as

$$\Phi(\mathbf{x}) = [\sqrt{\lambda_1}\phi_1(\mathbf{x}), \cdots, \sqrt{\lambda_k}\phi_i(\mathbf{x}), \cdots]^T \quad , \tag{4.7}$$

and define an inner product in $\mathbb{F}$ as

$$\left\langle [u_1, \cdots, u_i, \cdots]^T, [v_1, \cdots, v_i, \cdots]^T \right\rangle_{\mathbb{F}} = \sum_{i=1}^{\infty} u_i v_i \quad , \tag{4.8}$$

then (4.6) becomes

$$K(\mathbf{x}, \mathbf{z}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle_{\mathbb{F}} \quad . \tag{4.9}$$

The function $\Phi(\mathbf{x}) \in \mathbb{F}$ is sometimes referred to as the Mercer features. Equation (4.9) displays a nice property of Mercer kernels: a Mercer kernel implicitly defines a nonlinear mapping Φ such that the kernel computes the inner product in the space to which Φ maps. Therefore a Mercer kernel enables a classifier, in the form of (4.2), to operate on Mercer features (which usually reside in a space with dimension much higher than that of the input space) without explicitly evaluating the Mercer features (which is computationally very expensive). The following theorem illustrates the relationship between the PDFCs and Mercer features.

Theorem 4.13. *Given n positive definite reference functions, $a^k : \mathbb{R} \to [0, 1]$, $k = 1, \cdots, n$, and a compact set $\mathbb{X} \subset \mathbb{R}^n$, we define a Mercer kernel $K(\mathbf{x}, \mathbf{z}) = \prod_{k=1}^{n} a^k(x_k -$*

z_k) *where* $\mathbf{x} = [x_1, \cdots, x_n]^T, \mathbf{z} = [z_1, \cdots, z_n]^T \in \mathbb{X}$. *Let* $\mathbb{F}$ *be an* l_2 *space,* $\Phi : \mathbb{X} \to \mathbb{F}$ *be the nonlinear mapping given by (4.7), and* $\langle \cdot, \cdot \rangle_{\mathbb{F}}$ *be an inner product in* $\mathbb{F}$ *defined by (4.8). Given a set of points* $\{\mathbf{z}_1, \cdots, \mathbf{z}_m\} \subset \mathbb{X}$, *we define a subspace* $\mathbb{W} \subset \mathbb{F}$ *as* $\mathbb{W} = \text{Span}\{\Phi(\mathbf{z}_1), \cdots, \Phi(\mathbf{z}_m)\}$, *and a function space* $\mathbb{H}$ *on* $\mathbb{F}$ *as* $\mathbb{H} = \{h : h(\mathbf{u}) = \text{sign}(\langle \mathbf{w}, \mathbf{u} \rangle_{\mathbb{F}} + b_0),\ \mathbf{w} \in \mathbb{W},\ \mathbf{u} \in \mathbb{F},\ b_0 \in \mathbb{R}\}$. *Then we have the following results:*

1. *For any* $g \in \mathbb{H}$, *there exists a PDFC with* $a^k,\ k = 1, \cdots, n$, *as reference functions such that the decision rule,* f, *of the PDFC satisfies* $f(\mathbf{x}) = g(\Phi(\mathbf{x}))$, $\forall \mathbf{x} \in \mathbb{X}$.

2. *For any PDFC using* $a^k,\ k = 1, \cdots, n$, *as reference functions, if* $\mathbf{z}_j$ *contains location parameters of the IF-part membership functions associated with the* j*th fuzzy rule for* $j = 1, \cdots, m$ *(as defined in Corollary 4.6), then there exists* $g \in \mathbb{H}$ *such that the decision rule,* f, *of the PDFC satisfies* $f(\mathbf{x}) = g(\Phi(\mathbf{x}))$, $\forall \mathbf{x} \in \mathbb{X}$.

Proof:

1. Given $g \in \mathbb{H}$, we have $g(\mathbf{u}) = \text{sign}(\langle \mathbf{w}, \mathbf{u} \rangle_{\mathbb{F}} + b_0)$. Since $\mathbf{w} \in \mathbb{W}$, it can be written as a linear combination of $\Phi(\mathbf{z}_j)$'s, i.e., $\mathbf{w} = \sum_{j=1}^{m} b_j \Phi(\mathbf{z}_j)$. Thus $g(\mathbf{u})$ becomes

$$\begin{aligned} g(\mathbf{u}) &= \text{sign}\left(\left\langle \sum_{j=1}^{m} b_j \Phi(\mathbf{z}_j), \mathbf{u} \right\rangle_{\mathbb{F}} + b_0 \right) \\ &= \text{sign}\left(\sum_{j=1}^{m} b_j \left\langle \Phi(\mathbf{z}_j), \mathbf{u} \right\rangle_{\mathbb{F}} + b_0 \right) \quad . \end{aligned}$$

   Now we can define a PDFC using $a^k,\ k = 1, \cdots, n$, as reference functions. For $j = 1, \cdots, m$, let $\mathbf{z}_j$ contain location parameters of the IF-part membership functions associated with the jth fuzzy rule (as defined in Corollary 4.6), and b_j be the

THEN-part of the jth fuzzy rule. The THEN-part of Rule 0 is b_0. Then from (4.2) and (4.9), the decision rule is

$$\begin{aligned} f(\mathbf{x}) &= \text{sign}\left(\sum_{j=1}^{m} b_j K(\mathbf{x}, \mathbf{z}_j) + b_0\right) \\ &= \text{sign}\left(\sum_{j=1}^{m} b_j \left\langle \Phi(\mathbf{x}), \Phi(\mathbf{z}_j) \right\rangle_{\mathbb{F}} + b_0\right) \end{aligned}$$

Clearly, $f(\mathbf{x}) = g(\Phi(\mathbf{x}))$, $\forall \mathbf{x} \in \mathbb{X}$.

2. For a PDFC described in the theorem, let b_j be the THEN-part of the jth fuzzy rule, and b_0 be the THEN-part of Rule 0. Then from (4.2) and (4.9), the decision rule is

$$\begin{aligned} f(\mathbf{x}) &= \text{sign}\left(\sum_{j=1}^{m} b_j \left\langle \Phi(\mathbf{x}), \Phi(\mathbf{z}_j) \right\rangle_{\mathbb{F}} + b_0\right) \\ &= \text{sign}\left(\left\langle \sum_{j=1}^{m} b_j \Phi(\mathbf{z}_j), \Phi(\mathbf{x}) \right\rangle_{\mathbb{F}} + b_0\right) \quad . \end{aligned}$$

Let $\mathbf{w} = \sum_{j=1}^{m} b_j \Phi(\mathbf{z}_j)$ and $g(\mathbf{u}) = \text{sign}(\langle \mathbf{w}, \mathbf{u} \rangle_{\mathbb{F}} + b_0)$, then $g \in \mathbb{H}$ and $f(\mathbf{x}) = g(\Phi(\mathbf{x}))$, $\forall \mathbf{x} \in \mathbb{X}$.

This completes the proof. □

Remark 4.14. *The compactness of the input domain $\mathbb{X}$ is required for purely theoretical reason: it ensures that the expansion (4.6) can be written in a form of countable sum, thus the nonlinear mapping (4.7) can be defined. In practice, we don't need to worry about it provided that all input features (both training and testing) are within certain*

range (which can be satisfied via data preprocessing). Consequently, it is reasonable to assume that $\mathbf{z}_j$ *is also in* $\mathbb{X}$ *for* $j = 1, \cdots, m$ *because this essentially requires that all fuzzy rule "patches" center inside the input domain.*

Remark 4.15. *Since* $g(\mathbf{u}) = \text{sign}(\langle \mathbf{w}, \mathbf{u} \rangle_{\mathbb{F}} + b) = 0$ *defines a hyperplane in* $\mathbb{F}$*, Theorem 4.13 relates the decision boundary of a PDFC in* $\mathbb{X}$ *to a hyperplane in* $\mathbb{F}$*. The theorem implies that given any hyperplane in* $\mathbb{F}$*, if its orientation (normal direction pointed by* $\mathbf{w}$*) is a linear combination of vectors that have preimage (under* Φ*) in* $\mathbb{X}$*, then the hyperplane transforms to a decision boundary of a PDFC. Conversely, given a PDFC, one can find a hyperplane in* $\mathbb{F}$ *that transforms to the decision boundary of the given PDFC. Therefore, we can alternatively consider the decision boundary of a PDFC as a hyperplane in the feature space* $\mathbb{F}$*, which corresponds to a nonlinear decision boundary in* $\mathbb{X}$*. Constructing a PDFC is then converted to finding a hyperplane in* $\mathbb{F}$*.*

Remark 4.16. *A hyperplane in* $\mathbb{F}$ *is defined by its normal direction* $\mathbf{w}$ *and the distance to the origin, which is determined by* b *for fixed* $\mathbf{w}$*. According to the proof of Theorem 4.13,* $\mathbf{w}$ *and* b *are defined as* $\mathbf{w} = \sum_{j=1}^{m} b_j \Phi(\mathbf{z}_j)$ *and* $b = b_0$*, respectively, where* $\{\mathbf{z}_1, \cdots, \mathbf{z}_m\} \subset \mathbb{X}$ *is the set of location parameters of the IF-part fuzzy rules, and* $\{b_0, \cdots, b_m\} \subset \mathbb{R}$ *is the set of constants in the THEN-part fuzzy rules. This implies that the IF-part and THEN-part of fuzzy rules play different roles in modeling the hyperplane. The IF-part parameters,* $\{\mathbf{z}_1, \cdots, \mathbf{z}_m\}$*, defines a set of feasible orientations,* $\mathbb{W} = \text{Span}\{\Phi(\mathbf{z}_1), \cdots, \Phi(\mathbf{z}_m)\}$*, of the hyperplane. The THEN-part parameters* $\{b_1, \cdots, b_m\}$ *select an orientation,* $\sum_{j=1}^{m} b_j \Phi(\mathbf{z}_j)$*, from* $\mathbb{W}$*. The distance to the origin is then determined by the THEN-part of Rule* 0*, i.e.,* $b = b_0$*.*

4.5 An SVM Approach to Build Positive Definite Fuzzy Classifiers

A PDFC with n inputs and m, which is unknown, fuzzy rules is parameterized by n, possibly different, positive definite reference functions ($a^k : \mathbb{R} \to [0,1]$, $k = 1, ...n$), a set of location parameters ($\{\mathbf{z}_1, \cdots, \mathbf{z}_m\} \subset \mathbb{X}$) for the membership functions of the IF-part fuzzy rules, and a set of real numbers ($\{b_0, \cdots, b_m\} \subset \mathbb{R}$) for the constants in the THEN-part fuzzy rules. Which reference functions to choose is an interesting research topic by itself [74]. But it is out of the scope of this thesis. Here we assume that the reference functions $a^i : \mathbb{R} \to [0,1],\ i = 1, \cdots, n$ are predetermined. So the remaining question is how to find a set of fuzzy rules ($\{\mathbf{z}_1, \cdots, \mathbf{z}_m\}$ and $\{b_0, \cdots, b_m\}$) from the given training samples $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l)\} \subset \mathbb{X} \times \{+1, -1\}$ so that the PDFC has good generalization.

As given in (4.3), for a PDFC, a Mercer kernel can be constructed from the positive definite reference functions. The kernel implicitly defines a nonlinear mapping Φ that maps $\mathbb{X}$ into a kernel-induced feature space $\mathbb{F}$. Theorem 4.13 states that the decision rule of a PDFC can be viewed as a hyperplane in $\mathbb{F}$. Therefore, the original question transforms to: given training samples $\{(\Phi(\mathbf{x}_1), y_1), \cdots, (\Phi(\mathbf{x}_l), y_l)\} \subset \mathbb{F} \times \{+1, -1\}$, how to find a separating hyperplane in $\mathbb{F}$ that yields good generalization, and how to extract fuzzy rules from the obtained optimal hyperplane. We have seen in Section 4.2 that the SVM algorithm finds a separating hyperplane (in the input space or the kernel induced feature space) with good generalization by reducing the empirical risk and, at the same time, controlling the hyperplane margin. Thus we can use the SVM algorithm to find

an optimal hyperplane in $\mathbb{F}$. Once we get such a hyperplane, fuzzy rules can be easily extracted. The whole procedure is described by the following algorithm.

Algorithm 4.1. SVM Learning for PDFC

Inputs: Positive definite reference functions $a^k(x_k)$, $k = 1, \cdots, n$, associated with n input variables, and a set of training samples $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l)\}$.

Outputs: A set of fuzzy rules parameterized by $\mathbf{z}_j$, b_j, and m. $\mathbf{z}_j$ $(j = 1, \cdots, m)$ contains the location parameters of the IF-part membership functions of the jth fuzzy rule, b_j $(j = 0, \cdots, m)$ is the THEN-part constant of the jth fuzzy rule, and $m + 1$ is the number of fuzzy rules.

Steps:

1 construct a Mercer kernel, K, from the given positive definite reference functions according to (4.3).

2 construct an SVM to get a decision rule of the form (3.10):

  1) assign some positive number to C, and solve the quadratic program defined by (3.9) to get the Lagrange multipliers α.

  2) find b (details can be found in, for example, [11]).

3 extracting fuzzy rules from the decision rule of the SVM:

  $b_0 \leftarrow b$

  $j \leftarrow 1$

  FOR $i = 1$ TO l

    IF $\alpha_i > 0$

      $\mathbf{z}_j \leftarrow \mathbf{x}_i$

$b_j \leftarrow y_i \alpha_i$

$j \leftarrow j + 1$

END

END

$m \leftarrow j - 1$

It is straightforward to check that the decision rule of the resulting PDFC is identical to (3.10).

Once reference functions are fixed, the only free parameter in the above algorithm is C. According to the optimization criterion in (3.5), C weights the classification error versus the upper bound on the VC dimension. Another way of interpreting C is that it affects the sparsity of α (the number of nonzero entries in α) [8]. Unfortunately, there is no general rule for choosing C. Typically, a range of values of C should be tried before the best one can be selected.

The above learning algorithm has several nice properties:

- The shape of the reference functions and C parameter are the only prior information needed by the algorithm.

- The algorithm automatically generates a set of fuzzy rules. The number of fuzzy rules is irrelevant to the dimension of the input space. It equals the number of nonzero Lagrange multipliers. In this sense, the "curse of dimensionality" is avoided. In addition, due to the sparsity of α, the number of fuzzy rules is usually much less than the number of training samples.

- Each fuzzy rule is parameterized by a training sample $(\mathbf{x}_j, y_j)$ and the associated nonzero Lagrange multiplier α_j where $\mathbf{x}_j$ specifies the location of the IF-part membership functions, and $y_j\alpha_j$ gives the THEN-part constant.

- The global solution for the optimization problem can always be found efficiently because of the convexity of the objective function and of the feasible region. Algorithms designed specifically for the quadratic programming problems in SVMs make large-scale training (for example $200,000$ samples with $40,000$ input variables) practical [55, 56, 82]. The computational complexity of classification operation is determined by the cost of kernel evaluation and the number of support vectors.

- Since the goal of optimization is to lower an upper bound on the expected risk (not just the empirical risk), the resulting PDFC usually has good generalization.

4.6 Discussions

In the literature, it is well-known that a Gaussian RBF network can be trained via support vector learning using a Gaussian RBF kernel [92]. While the functional equivalence between fuzzy inference systems and Gaussian RBF networks is established in [53] where the membership functions within each rule must be Gaussian functions with identical variance. So connection between such fuzzy systems and SVMs with Gaussian RBF kernels can be established. The following discussion compares the kernels defined by PDFCs and RBF kernels commonly used in SVMs.

The kernels of PDFCs are constructed from positive definite reference functions. These kernels are translation invariant, symmetric with respect to a set of orthogonal axes, and tailing off gradually. In this sense, they appear to be very similar to the general RBF kernels [36]. In fact, the Gaussian reference function defines the Gaussian RBF kernel. However, in general, the kernels of PDFCs are not RBF kernels. According to the definition, an RBF kernel, $K(\mathbf{x}, \mathbf{z})$, depends only on the norm of $\mathbf{x} - \mathbf{z}$, i.e., $K(\mathbf{x}-\mathbf{z}) = K_{RBF}(\|\mathbf{x}-\mathbf{z}\|)$. It can be shown that for a kernel, $K(\mathbf{x}, \mathbf{z})$, defined by (4.3) using symmetric triangle, Cauchy, Laplace, hyperbolic secant, or squared sinc reference functions (even with identical d for all input variables), there exists $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{z}_1$, and $\mathbf{z}_2$ such that $\|\mathbf{x}_1 - \mathbf{z}_1\| = \|\mathbf{x}_2 - \mathbf{z}_2\|$ and $K(\mathbf{x}_1, \mathbf{z}_1) \neq K(\mathbf{x}_2, \mathbf{z}_2)$. Moreover, a general RBF kernels (even if it is a Mercer kernel) may not be a PDFC kernel, i.e., it can not be in general decomposed as product of positive definite reference functions. It is worth noting that the kernel defined by symmetric triangle reference functions is identical to the B_n-splines (or order 1) kernel that is commonly used in the SVM literature [117].

Chapter 5

A Robust Image Similarity Measure Using Fuzzified Region Features

This chapter starts with an overview of the proposed approach. Then we describe image segmentation and fuzzy feature representation of an image in Section 5.2. A similarity measure is introduced in Section 5.3. Section 5.4 provides an algorithmic presentation of the resulting CBIR system.

5.1 Overview

Semantically precise image segmentation by an algorithm is very difficult [96, 119, 133]. However, a single glance is sufficient for human to identify circles, straight lines, and other complex objects in a collection of points and to produce a meaningful assignment between objects and points in the image. Although those points cannot always be assigned unambiguously to objects, human recognition performance is hardly affected. We can often identify the object of interest correctly even when its boundary is very blurry. This is probably because the prior knowledge of similar objects and images may provide powerful assistance for human in recognition. Unfortunately, this prior knowledge is usually unavailable to most of the current CBIR systems. However, we argue that a similarity measure allowing for blurry boundaries between regions may increase the performance of a region-based CBIR system. To improve the robustness of a region-based image retrieval system against segmentation-related uncertainties, which

always exist due to inaccurate image segmentation, we propose unified feature matching (UFM) scheme based on fuzzy logic theory.

Applying fuzzy processing techniques to CBIR has been extensively investigated in the literature. In [61], fuzzy logic is developed to interpret the overall color information of images. Nine colors that match human perceptual categories are chosen as features. Vertan *et al.* propose a fuzzy color histogram approach in [118]. A class of similarity distances is defined based on fuzzy logic operations. Our scheme is distinct from the above methods in two aspects:

- It is a region-based fuzzy feature matching approach. Segmentation-related uncertainties are viewed as blurring boundaries between segmented regions. Instead of a feature vector, we represent each region as a multidimensional fuzzy set, named fuzzy feature, in the feature space of color, texture, and shape. Thus, each image is characterized by a class of fuzzy features. Fuzzy features naturally characterize the gradual transition between regions (blurry boundaries) within an image. It assigns weights, called *degrees of membership*, to every feature vectors in the feature space. As a result, a feature vector usually belongs to multiple regions with different degrees of membership as opposed to the classical region representation, in which a feature vector belongs to exactly one region.

- A novel image similarity measure, UFM measure, is derived from fuzzy set operations. The matching of two images is performed in three steps. First, each fuzzy feature of the query image is matched with all fuzzy features of the target

image in a *Winner Takes All* fashion. Then, each fuzzy feature of the target image is matched with all fuzzy features of the query image using the same strategy as in the previous step. Finally, overall similarity, given as the UFM measure, is calculated by properly weighting the results from the above two steps.

5.2 Image Segmentation and Representation

The building blocks for the UFM approach are segmented regions and the corresponding fuzzy features. In our system, the query image and all images in the database are first segmented into regions. Regions are then represented by multidimensional fuzzy sets in the feature space. The collection of fuzzy sets for all regions of an image constitutes the *signature* of the image.

5.2.1 Image Segmentation

Our system segments images based on color and spatial variation features using k-means algorithm [43], a very fast statistical clustering method. For general-purpose images such as the images in a photo library or the images on the World-Wide Web, precise object segmentation is nearly as difficult as computer semantics understanding. However, semantically-precise segmentation is not crucial to our system because our UFM approach is insensitive to inaccurate segmentation.

To segment an image, the system first partitions the image into small blocks. A feature vector is then extracted for each block. The block size is chosen to compromise between texture effectiveness and computation time. Smaller block size may preserve more texture details but increase the computation time as well. Conversely, increasing

the block size can reduce the computation time but lose texture information and increase the segmentation coarseness. In our current system, each block has 4×4 pixels. The size of the images in our database is either 256×384 or 384×256. Therefore each image corresponds to 6144 feature vectors. Each feature vector, $\mathbf{f}_i$, consists of six features, i.e., $\mathbf{f}_i \in \mathbb{R}^6$, $1 \leq i \leq 6144$. Three of them are the average color components in a 4×4 block. We use the well-known LUV color space, where L encodes luminance, and U and V encode color information (chrominance). The other three represent energy in the high frequency bands of the wavelet transforms [25], that is, the square root of the second order moment of wavelet coefficients in high frequency bands.

To obtain these moments, a Daubechies-4 wavelet transform is applied to the L component of the image. After a one-level wavelet transform, a 4×4 block is decomposed into four frequency bands: the LL, LH, HL, and HH bands. Each band contains 2×2 coefficients. Without loss of generality, suppose the coefficients in the HL band are $\{c_{k,l}, c_{k,l+1}, c_{k+1,l}, c_{k+1,l+1}\}$. One feature is

$$f = \left(\frac{1}{4} \sum_{i=0}^{1} \sum_{j=0}^{1} c_{k+i,l+j}^2 \right)^{\frac{1}{2}}.$$

The other two features are computed similarly from the LH and HH bands. The motivation for using the features extracted from high frequency bands is that they reflect texture properties. Moments of wavelet coefficients in various frequency bands have been shown to be effective for representing texture [111]. The intuition behind this is that

coefficients in different frequency bands show variations in different directions. For example, the HL band shows activities in the horizontal direction. An image with vertical strips thus has high energy in the HL band and low energy in the LH band.

The k-means algorithm is used to cluster the feature vectors into several classes with every class corresponding to one region in the segmented image, i.e., for an image with the set of feature vectors $\mathbf{F} = \{\mathbf{f}_i \in \mathbb{R}^6 : 1 \leq i \leq 6144\}$, $\mathbf{F}$ is partitioned into C groups $\{\mathbf{F}_1, \cdots, \mathbf{F}_C\}$, and consequently, the image is segmented into C regions $\{\mathbf{R}_1, \cdots, \mathbf{R}_C\}$ with $\mathbf{R}_j \subset \mathbb{N}^2$ being the region corresponding to the feature set $\mathbf{F}_j$, $1 \leq j \leq C$. Because clustering is performed in the feature space, blocks in each cluster do not necessarily form a connected region in the images. This way, we preserve the natural clustering of objects in textured images and allow classification of textured images [65]. The k-means algorithm does not specify how many clusters to choose. We adaptively select the number of clusters C by gradually increasing C until a stop criterion is met. The average number of clusters for all images in the database changes in accordance with the adjustment of the stop criteria. As we will see in Section 8.1, the average number of clusters is closely related to segmentation-related uncertainty level, and hence affects the performance of the system.

After segmentation, three extra features are calculated for each region to describe shape properties. They are normalized inertia [37] of order 1 to 3. For a region $\mathbf{R}_j \subset \mathbb{N}^2$ in the image plane, which is a finite set, the normalized inertia of order γ is given as

$$I_{(\mathbf{R}_j,\gamma)} = \frac{\sum_{(x,y):(x,y)\in\mathbf{R}_j}[(x-\hat{x})^2 + (y-\hat{y})^2]^{\frac{\gamma}{2}}}{\mathrm{V}(\mathbf{R_j})^{1+\frac{\gamma}{2}}},$$

where $(\hat{x}, \hat{y})$ is the centroid of $\mathbf{R}_j$, V($\mathbf{R}_\mathrm{j}$) is the volume of $\mathbf{R}_j$. The normalized inertia is invariant to scaling and rotation. The minimum normalized inertia is achieved by spheres. Denote the γth order normalized inertia of spheres as $\mathbf{I}_\gamma$. We define shape feature $\mathbf{h}_j$ of region $\mathbf{R}_j$ as $I_{(\mathbf{R}_j,\gamma)}$ normalized by $\mathbf{I}_\gamma$, i.e.,

$$\mathbf{h}_j = \left[\frac{I_{(\mathbf{R}_j,1)}}{\mathbf{I}_1}, \frac{I_{(\mathbf{R}_j,2)}}{\mathbf{I}_2}, \frac{I_{(\mathbf{R}_j,3)}}{\mathbf{I}_3} \right]^T .$$

5.2.2 Fuzzy Feature Representation of an Image

A segmented image can be viewed as a collection of regions, $\{\mathbf{R}_1, \cdots, \mathbf{R}_C\}$. Equivalently, in the feature space, the image is characterized by a collection of feature sets, $\{\mathbf{F}_1, \cdots, \mathbf{F}_C\}$, which form a partition of $\mathbf{F}$. We could use the feature set $\mathbf{F}_j$ to describe the region $\mathbf{R}_j$, and compute the similarity between two images based on $\mathbf{F}_j$'s. Representing regions by feature sets incorporates all the information available in the form of feature vectors, but it has two drawbacks:

- It is sensitive to segmentation-related uncertainties. For any feature vector in $\mathbf{F}$, under this region representation, it belongs to exactly one feature set. But, in general, image segmentation cannot be perfect. As a result, for many feature vectors, a unique decision between in and not in the feature set is impossible.

- The computational cost for similarity calculation is very high. Usually, the similarity measure for two images is calculated based on the distances (Euclidean distance is the one that is commonly used in many applications) between feature vectors

from different images. Therefore, for each image in the database, we need to compute $\frac{6144\times6145}{2}$ such distances. Even with a rather conservative assumption, one CPU clock cycle per distance, it takes about half an hour just to compute the Euclidean distances for all $60,000$ images in our database on a 700MHz PC. This amount of time is certainly too much for system users to tolerate.

In an improved region representation [64], which mitigates the above drawbacks, each region ($\mathbf{R}_j$) is represented by the center ($\widehat{\mathbf{f}}_j$) of the corresponding feature set ($\mathbf{F}_j$) with $\widehat{\mathbf{f}}_j$ defined as

$$\widehat{\mathbf{f}}_j = \frac{\sum_{\mathbf{f}\in\mathbf{F}_j}\mathbf{f}}{\mathrm{V}(\mathbf{F}_\mathrm{j})}, \tag{5.1}$$

which is essentially the mean of all elements of $\mathbf{F}_j$, and in general may not be an element of $\mathbf{F}_j$. While averaging over all features in a feature set decreases the impact of inaccurate segmentation, at the same time, lots of useful information is also submerged in the smoothing process because a set of feature vectors are mapped to a single feature vector. Moreover, the segmentation-related uncertainties are not explicitly expressed in this region representation.

Representing regions by fuzzy features, to some extent, combines the advantages and avoids the drawbacks of both region representations mentioned above. In this representation, each region is associated with a fuzzy feature that assigns a value (between 0 and 1) to each feature vector in the feature space. The value, named *degree of membership*, illustrates the degree of wellness that a corresponding feature vector characterizes the region, and thus models the segmentation-related uncertainties. In Section 5.3, we

will show that this representation leads to a computationally efficient region matching scheme if appropriate membership functions are selected.

A fuzzy feature $\mathbf{F}$ on the feature space $\mathbb{R}^6$ is defined by a mapping $\mu_{\mathbf{F}} : \mathbb{R}^6 \rightarrow [0, 1]$ named the *membership function*. For any feature vector $\mathbf{f} \in \mathbb{R}^6$, the value of $\mu_{\mathbf{F}}(\mathbf{f})$ is called the degree of membership of $\mathbf{f}$ to the fuzzy feature $\mathbf{F}$ (or, in short, the degree of membership to $\mathbf{F}$). A value closer to 1 for $\mu_{\mathbf{F}}(\mathbf{f})$ means more representative the feature vector $\mathbf{f}$ is to the corresponding region. For a fuzzy feature $\mathbf{F}$, there is a smooth transition for the degree of membership to $\mathbf{F}$ besides the hard cases $\mathbf{f} \in \mathbf{F}$ ($\mu_{\mathbf{F}}(\mathbf{f}) = 1$) and $\mathbf{f} \notin \mathbf{F}$ ($\mu_{\mathbf{F}}(\mathbf{f}) = 0$). It is clear that a fuzzy feature degenerates to a conventional feature set if the range of $\mu_{\mathbf{F}}$ is $\{0, 1\}$ instead of $[0, 1]$ ($\mu_{\mathbf{F}}$ is then called the *characteristic function* of the feature set).

Building or choosing a proper membership function is an application dependent problem. Some most commonly used prototype membership functions are cone, exponential, and Cauchy functions [46]. Two factors are considered when we select the membership function for our system: retrieval accuracy and computational intensity for evaluating a membership function. For different membership functions, although the discrepancies among the efforts of computing degrees of membership are small, it is not negligible for large-sized image databases as, in a retrieval process, it is magnified by the product of the number of regions in the query image and the number of images in the database. As shown in Section 8.1.4, under proper parameters, the cone, exponential, and Cauchy functions can capture the uncertainties in feature vectors almost equally well, which is reflected by retrieval accuracies of the resulting systems. But computational

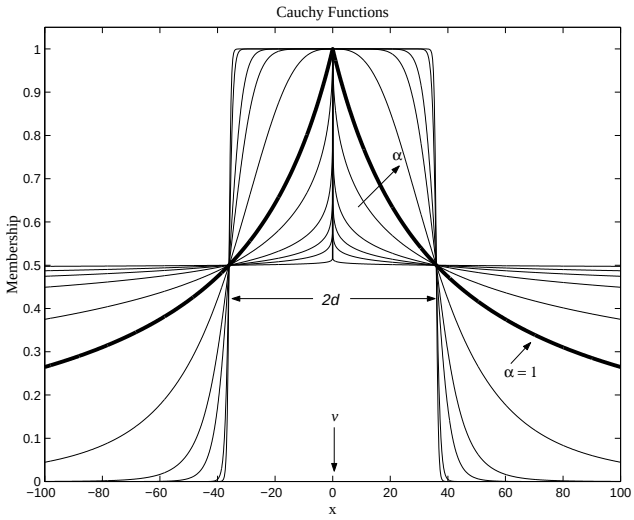


Fig. 5.1. Cauchy functions in $\mathbb{R}^1$.

intensities vary. As a result, we pick the Cauchy function due to its good expressiveness and high computational efficiency. A detailed comparison of all three membership functions are given in Section 8.1.4.

The Cauchy function, $\mathcal{C} : \mathbb{R}^k \rightarrow [0, 1]$, is defined as

$$\mathcal{C}(\mathbf{x}) = \frac{1}{1 + \left(\frac{\|\mathbf{x}-\mathbf{v}\|}{d}\right)^{\alpha}} \tag{5.2}$$

where $\mathbf{v} \in \mathbb{R}^k$, d and $\alpha \in \mathbb{R}$, $d > 0$, $\alpha \geq 0$. $\mathbf{v}$ is the *center location* (point) of the function (or called the center location of the fuzzy set), d represents the *width* ($\|\mathbf{x} - \mathbf{v}\|$ for $\mathcal{C}(\mathbf{x}) = 0.5$) of the function, and α determines the *shape* (or *smoothness*) of the function. Collectively, d and α portray the *grade of fuzziness* of the corresponding fuzzy feature. For fixed d, the grade of fuzziness increases as α decreases. If α is fixed, the grade of

fuzziness increases with the increasing of d. Figure 5.1 illustrates Cauchy functions in $\mathbb{R}$ with $v = 0$, $d = 36$, and α varying from 0.01 to 100. As we can see, the Cauchy function approaches the characteristic function of open interval $(-36, 36)$ when α goes to positive infinity. When α equals 0, the degree of membership for any element in $\mathbb{R}$ (except 0, whose degree of membership is always 1 in this example) is 0.5.

Accordingly, the region $\mathbf{R}_j$ is represented by fuzzy feature $\mathbf{F}_j$ whose membership function, $\mu_{\mathbf{F}_j} : \mathbb{R}^6 \rightarrow [0, 1]$, is defined as

$$\mu_{\mathbf{F}_j}(\mathbf{f}) = \frac{1}{1 + \left(\frac{\|\mathbf{f} - \widehat{\mathbf{f}}_j\|}{d_f}\right)^{\alpha}} \tag{5.3}$$

where

$$d_f = \frac{2}{C(C-1)} \sum_{i=1}^{C-1} \sum_{k=i+1}^{C} \|\widehat{\mathbf{f}}_i - \widehat{\mathbf{f}}_k\|$$

is the average distance between cluster centers, $\widehat{\mathbf{f}}_i$'s, defined by (5.1). An interesting property intrinsic to membership function (5.3) is that the farther a feature vector moves away from the cluster center, the lower its degree of membership to the fuzzy feature. At the same time, its degrees of membership to some other fuzzy features may be increasing. This nicely describes the gradual transition of region boundaries.

As stated in Section 5.2.1, the shape properties of region $\mathbf{R}_j$ is described by shape feature $\mathbf{h}_j$. Considering the impacts of inaccurate segmentation on the shapes of regions, it is reasonable to use fuzzy sets to illustrate shape properties. Thus, for region $\mathbf{R}_j$, the shape feature $\mathbf{h}_j$ is extended to a fuzzy set $\mathbf{H}_j$ with membership function,

$\mu_{\mathbf{H}_j} : \mathbb{R}^3 \rightarrow [0,1]$, defined as

$$\mu_{\mathbf{H}_j}(\mathbf{h}) = \frac{1}{1 + \left(\frac{\|\mathbf{h}-\mathbf{h}_j\|}{d_h}\right)^{\alpha}} \tag{5.4}$$

where

$$d_h = \frac{2}{C(C-1)} \sum_{i=1}^{C-1} \sum_{k=i+1}^{C} \|\mathbf{h}_i - \mathbf{h}_k\|$$

is the average distance between shape features. The experiments show that the performance changes insignificantly when α is in the interval $[0.9, 1.2]$, but degrades rapidly outside the interval. This is probably because, as α decreases, the Cauchy function becomes sharper within its center region ($[-d, d]$ for the example in Figure 5.1) and flatter outside. As a result, many useful feature vectors within that region are likely to be overlooked since their degrees of membership become smaller. Conversely, when α is large, the Cauchy function becomes flat within the center region. Consequently, the noise feature vectors in that region are likely to be selected as their degrees of membership are high. We set $\alpha = 1$ in both (5.3) and (5.4) based on the experimental results in Section 8.1.4.

For an image with regions $\mathbf{R}_j$, $1 \leq j \leq C$, $(\mathcal{F}, \mathcal{H})$ is named the *fuzzy feature representation* (or *signature*) of the image, where $\mathcal{F} = \{\mathbf{F}_j : 1 \leq j \leq C, j \in \mathbb{N}\}$ with $\mathbf{F}_j$ defined by (5.3), $\mathcal{H} = \{\mathbf{H}_j : 1 \leq j \leq C, j \in \mathbb{N}\}$ with $\mathbf{H}_j$ defined by (5.4). The color and texture properties are characterized by $\mathcal{F}$, while the shape properties are captured by $\mathcal{H}$.

5.2.3 An Algorithmic View

The image segmentation and fuzzy feature representation process can be summarized as follows. $\epsilon_1 > 0$ and $\epsilon_2 > 0$ are given stop criteria. The input is an image in raw format. The outputs is the signature of the image, $(\mathcal{F}, \mathcal{H})$, which is characterized by $\hat{\mathbf{f}}_j \in \mathbb{R}^6$ (center location) and $d_f > 0$ (width) of color/texture fuzzy features, and $\mathbf{h}_j \in \mathbb{R}^3$ (center location) and $d_h > 0$ (width) of shape fuzzy features. $j = 1 \ldots C$, C is the number of regions.

Algorithm 5.1. Image Segmentation and Fuzzy Features Extraction

partition the image into B 4×4 blocks
FOR $i = 1$ TO B
    extract feature vector $\mathbf{f}_i$ for block i
END
$k \leftarrow 2$, $D[1] \leftarrow 0$
WHILE $k \leq M$
    group $\{\mathbf{f}_i : 1 \leq i \leq B\}$ into k clusters using the k-means algorithm
    $C \leftarrow k$
    FOR $j = 1$ TO C
        compute the mean, $\hat{\mathbf{f}}_j$, for cluster j
    END
    $D[k] \leftarrow \sum_{i=1}^{B} \min_{1 \leq j \leq C} \|\mathbf{f}_i - \hat{\mathbf{f}}_j\|^2$
    IF $D[k] < \epsilon_1$ OR $D[k] - D[k-1] < \epsilon_2$
        $k \leftarrow M + 1$

```
    ELSE
        k ← k + 1
    END
END
FOR j = 1 TO C
    compute shape feature h_j for region j
END
d_f ← 0, d_h ← 0
FOR i = 1 TO C − 1
    FOR j = i + 1 TO C
        d_f ← d_f + ‖f̂_i − f̂_j‖
        d_h ← d_h + ‖h_i − h_j‖
    END
END
d_f ← 2d_f / (C(C−1)), d_h ← 2d_h / (C(C−1))
```

5.3 Unified Feature Matching

In this section, we describe the unified feature matching (UFM) scheme which characterizes the resemblance between images by integrating properties of all regions in the images. Based upon fuzzy feature representation of images, characterizing the similarity between images becomes an issue of finding similarities between fuzzy features. We first introduce a *fuzzy similarity measure* for two regions. The result is then extended

to construct a *similarity vector* which includes the region-level similarities for all regions in two images. Accordingly, a *similarity vector pair* is defined to illustrate the resemblance between two images. Finally, the UFM measure maps a similarity vector pair to a scalar quantity, within the real interval $[0, 1]$, which quantifies the overall image-to-image similarity.

5.3.1 Similarity Between Regions: Fuzzy Similarity Measure

Considering the fuzzy feature representation of images, the similarity between two regions can be captured by a fuzzy similarity measure of the corresponding fuzzy features (fuzzy sets). In the classical set theory, there are many definitions of similarity measure for sets. For example, a similarity measure of set $\mathbf{A}$ and $\mathbf{B}$ can be defined as the maximum value of the characteristic function of $\mathbf{A} \cap \mathbf{B}$, i.e., if they have common elements then the similarity measure is 1 (most similar), otherwise 0 (least similar). If $\mathbf{A}$ and $\mathbf{B}$ are finite sets, another definition is $\frac{\mathrm{V}(\mathbf{A} \cap \mathbf{B})}{\sqrt{\mathrm{V}(\mathbf{A})\mathrm{V}(\mathbf{B})}}$, meaning the more elements they have in common, the more similar they are. Almost all similarity measures for conventional sets have their counterparts in fuzzy domain [4]. Taking the computational complexity into account, in this paper, we use a definition extended from the first definition mentioned above.

Before giving the formal definition of the fuzzy similarity measure for two fuzzy sets, we first define elementary set operations, intersection and union, for fuzzy sets. Let $\mathbf{A}$ and $\mathbf{B}$ be fuzzy sets defined on $\mathbb{R}^k$ with corresponding membership functions $\mu_{\mathbf{A}} : \mathbb{R}^k \rightarrow [0, 1]$ and $\mu_{\mathbf{B}} : \mathbb{R}^k \rightarrow [0, 1]$, respectively. The intersection of $\mathbf{A}$ and $\mathbf{B}$, denoted by $\mathbf{A} \cap \mathbf{B}$, is a fuzzy set on $\mathbb{R}^k$ with membership function, $\mu_{\mathbf{A} \cap \mathbf{B}} : \mathbb{R}^k \rightarrow [0, 1]$,

defined as

$$\mu_{\mathbf{A}\cap\mathbf{B}}(\mathbf{x}) = \min[\mu_{\mathbf{A}}(\mathbf{x}), \mu_{\mathbf{B}}(\mathbf{x})]. \tag{5.5}$$

The union $\mathbf{A}$ and $\mathbf{B}$, denoted by $\mathbf{A} \cup \mathbf{B}$, is a fuzzy set on $\mathbb{R}^k$ with membership function, $\mu_{\mathbf{A}\cup\mathbf{B}} : \mathbb{R}^k \to [0, 1]$, defined as

$$\mu_{\mathbf{A}\cup\mathbf{B}}(\mathbf{x}) = \max[\mu_{\mathbf{A}}(\mathbf{x}), \mu_{\mathbf{B}}(\mathbf{x})]. \tag{5.6}$$

Note that there exists different definitions of intersection and union, the above definitions are computationally simplest [4].

The *fuzzy similarity measure* for fuzzy sets $\mathbf{A}$ and $\mathbf{B}$, $\mathcal{S}(\mathbf{A}, \mathbf{B})$, is given by

$$\mathcal{S}(\mathbf{A}, \mathbf{B}) = \sup_{\mathbf{x}\in\mathbb{R}^k} \mu_{\mathbf{A}\cap\mathbf{B}}(\mathbf{x}). \tag{5.7}$$

It is clear that $\mathcal{S}(\mathbf{A}, \mathbf{B})$ is always within the real interval $[0, 1]$ with a larger value denoting a higher degree of similarity between $\mathbf{A}$ and $\mathbf{B}$. For the fuzzy sets defined by Cauchy functions, as in (5.2), calculating the fuzzy similarity measure according to (5.7) is relatively simple. This is because the Cauchy function is unimodal, and therefore the maximum of (5.5) can only occur on the line segments connecting the center locations of two functions. It is not hard to show that for fuzzy sets $\mathbf{A}$ and $\mathbf{B}$ on $\mathbb{R}^k$ with Cauchy membership functions

$$\mu_{\mathbf{A}}(\mathbf{x}) = \frac{1}{1 + \left(\frac{\|\mathbf{x}-\mathbf{u}\|}{d_a}\right)^{\alpha}}$$

and

$$\mu_{\mathbf{B}}(\mathbf{x}) = \frac{1}{1 + \left(\frac{\|\mathbf{x}-\mathbf{v}\|}{d_b}\right)^{\alpha}} \quad ,$$

the fuzzy similarity measure for $\mathbf{A}$ and $\mathbf{B}$, which is defined by (5.7), can be equivalently written as

$$\mathcal{S}(\mathbf{A}, \mathbf{B}) = \frac{(d_a + d_b)^{\alpha}}{(d_a + d_b)^{\alpha} + \|\mathbf{u} - \mathbf{v}\|^{\alpha}} \quad . \tag{5.8}$$

5.3.2 Fuzzy Feature Matching

It is clear that the resemblance of two images is conveyed through the similarities between regions from both images. Thus it is desirable to construct the image-level similarity using region-level similarities. Since image segmentation is usually not perfect, a region in one image could correspond to several regions in another image. For example, a segmentation algorithm may segment an image of dog into two regions: the dog and the background. The same algorithm may segment another image of dog into five regions: the body of the dog, the front leg(s) of the dog, the rear leg(s) of the dog, the background grass, and the sky. There are similarities between the dog in the first image and the body, the front leg(s), or the rear leg(s) of the dog in the second image. The background of the first image is also similar to the background grass or the sky of the second image. However, the dog in the first image is unlikely to be similar to the background grass and sky in the second image.

Using fuzzy feature representation, these similarity observations can be expressed as:

- The similarity measure, given by (5.8), for the fuzzy feature of the dog in the first image and the fuzzy features of the dog body, front leg(s), **OR** rear leg(s) in the second image is high (e.g., close to 1).

- The similarity measure for the fuzzy feature of the background in the first image and the fuzzy features of the background grass **OR** sky in the second image is also high.

- The similarity measure for the fuzzy feature of the dog in the first image and the fuzzy feature of the background grass in the second image is small (e.g., close to 0). The similarity measure for the fuzzy feature of the dog in the first image and the fuzzy feature of the sky in the second image is also small.

Based on these qualitative illustrations, it is natural to think of the mathematical meaning of the word **OR**, i.e., the union operation. What we have described above is essentially the matching of a fuzzy feature with the union of some other fuzzy features. Based on this motivation, we construct the *similarity vector* for two classes of fuzzy sets through the following steps.

Let $\mathcal{A} = \{\mathbf{A}_i : 1 \leq i \leq C_a, i \in \mathbb{N}\}$, and $\mathcal{B} = \{\mathbf{B}_j : 1 \leq j \leq C_b, j \in \mathbb{N}\}$ denote two collections of fuzzy sets. First, for every $\mathbf{A}_i \in \mathcal{A}$, we define the similarity measure for it and $\mathcal{B}$ as

$$l_i^{\mathcal{B}} = \mathcal{S}(\mathbf{A}_i, \bigcup_{j=1}^{C_b} \mathbf{B}_j). \tag{5.9}$$

Combining $l_i^{\mathcal{B}}$'s together, we get a vector

$$\mathbf{l}^{\mathcal{B}} = [l_1^{\mathcal{B}}, l_2^{\mathcal{B}}, \cdots, l_{C_a}^{\mathcal{B}}]^T.$$

Similarly, for every $\mathbf{B}_j \in \mathcal{B}$, we define the similarity measure between it and $\mathcal{A}$ as

$$l_j^{\mathcal{A}} = \mathcal{S}(\mathbf{B}_j, \bigcup_{i=1}^{C_a} \mathbf{A}_i). \tag{5.10}$$

Combining $l_j^{\mathcal{A}}$'s together, we get a vector

$$\mathbf{l}^{\mathcal{A}} = [l_1^{\mathcal{A}}, l_2^{\mathcal{A}}, \cdots, l_{C_b}^{\mathcal{A}}]^T.$$

It is clear that $\mathbf{l}^{\mathcal{B}}$ describes the similarity between individual fuzzy features in $\mathcal{A}$ and all fuzzy features in $\mathcal{B}$. Likewise, $\mathbf{l}^{\mathcal{A}}$ illustrates the similarity between individual fuzzy features in $\mathcal{B}$ and all fuzzy features in $\mathcal{A}$. Thus we define a *similarity vector* for $\mathcal{A}$ and $\mathcal{B}$, denoted by $\mathbf{L}^{(\mathcal{A},\mathcal{B})}$, as

$$\mathbf{L}^{(\mathcal{A},\mathcal{B})} = \begin{bmatrix} \mathbf{l}^{\mathcal{B}} \\ \\ \mathbf{l}^{\mathcal{A}} \end{bmatrix},$$

which is a $C_a + C_b$ dimensional vector with values of all entries within the real interval $[0, 1]$.

It can be shown that if $\mathcal{A} = \mathcal{B}$ [1] then $\mathbf{L}^{(\mathcal{A},\mathcal{B})}$ contains all 1's. If a fuzzy set of $\mathcal{A}$ ($\mathcal{B}$) is quite different from all fuzzy sets of $\mathcal{B}$ ($\mathcal{A}$), in the sense that the distances between

[1] $\mathcal{A} = \mathcal{B}$ if and only if the membership functions of fuzzy sets in $\mathcal{A}$ are the same as those of fuzzy sets in $\mathcal{B}$.

their centers are much larger than their widths, the corresponding entry in $\mathbf{L}^{(\mathcal{A},\mathcal{B})}$ would be close to 0. Using the definition of the union of fuzzy sets, which is given by (5.6), equations (5.9) and (5.10) can be equivalently written as

$$l_i^{\mathcal{B}} = \max_{j=1,\cdots,C_b} \mathcal{S}(\mathbf{A}_i, \mathbf{B}_j), \tag{5.11}$$

$$l_j^{\mathcal{A}} = \max_{i=1,\cdots,C_a} \mathcal{S}(\mathbf{B}_j, \mathbf{A}_i). \tag{5.12}$$

Equations (5.11) and (5.12) shows that computing the similarity measure for $\mathbf{A}_i$ ($\mathbf{B}_j$) and $\mathcal{B}$ ($\mathcal{A}$) is equivalent to calculating the similarity measures for $\mathbf{A}_i$ ($\mathbf{B}_j$) and $\mathbf{B}_j$ ($\mathbf{A}_i$) with j taking integer values from 1 to C_b (i taking integer values from 1 to C_a), and then picking the maximum value, i.e., in a Winner Takes All fashion.

Let $(\mathcal{F}_q, \mathcal{H}_q)$ and $(\mathcal{F}_t, \mathcal{H}_t)$ be fuzzy feature representations for query image (q) and target image (t), respectively. The similarity between the query and target images is then captured by a *similarity vector pair* $(\mathbf{L}^{(\mathcal{F}_q,\mathcal{F}_t)}, \mathbf{L}^{(\mathcal{H}_q,\mathcal{H}_t)})$ where $\mathbf{L}^{(\mathcal{F}_q,\mathcal{F}_t)}$ depicts the similarity in colors and textures, and $\mathbf{L}^{(\mathcal{H}_q,\mathcal{H}_t)}$ describes the similarity in shapes. Within the similarity vectors, $\mathbf{l}^{\mathcal{F}_q}$ and $\mathbf{l}^{\mathcal{H}_q}$ refer to the similarity between the query image and regions of the target image. Likewise, $\mathbf{l}^{\mathcal{F}_t}$ and $\mathbf{l}^{\mathcal{H}_t}$ designate the similarity between the target image and regions of the query image.

5.3.3 The UFM Measure

Endeavoring to provide an overall image-to-image and intuitive similarity quantification, the UFM measure is defined as the summation of all the weighted entries of similarity vectors $\mathbf{L}^{(\mathcal{F}_q,\mathcal{F}_t)}$ and $\mathbf{L}^{(\mathcal{H}_q,\mathcal{H}_t)}$. We have discussed the methods of computing

similarity vectors in Sections 5.3.1 and 5.3.2. The problem is then converted to designing a weighting scheme. The UFM measure is computed in two stages. First, the inner products of similarity vectors $\mathbf{L}^{(\mathcal{F}_q,\mathbf{F}_t)}$ and $\mathbf{L}^{(\mathcal{H}_q,\mathcal{H}_t)}$ with weight vectors $\mathbf{w}_1$ and $\mathbf{w}_2$, respectively, are calculated. The results are then weighted by ρ_1 and ρ_2, and added up to give the UFM measure $m_{(q,t)}$.

There are many ways of choosing weight vectors $\mathbf{w}_1$ and $\mathbf{w}_2$. For example, in a *uniform weighting scheme* we assume every region being equally important. Thus all entries of $\mathbf{w}_1$ and $\mathbf{w}_2$ equal to $\frac{1}{C_q+C_t}$ where C_q (C_t) is the number of regions in the query (target) image. Such weight vectors favor the image with more regions because, in both $\mathbf{w}_1$ and $\mathbf{w}_2$, the summation of weights associated with the regions of the query (target) image is $\frac{C_q}{C_q+C_t}$ ($\frac{C_t}{C_q+C_t}$). If the regions within the same image are regarded as equally important, then the weights for entries of $\mathbf{l}^{\mathcal{F}_q}$ and $\mathbf{l}^{\mathcal{H}_q}$ ($\mathbf{l}^{\mathcal{F}_t}$ and $\mathbf{l}^{\mathcal{H}_t}$) can be chosen as $\frac{1}{2C_q}$ ($\frac{1}{2C_t}$). It is clear that regions from the image with less regions are allocated larger weights (if $C_q = C_t$ then the weights are identical to those under the uniform weighting scheme). We can also take the location of the regions into account, and assign higher weights to regions closer to the center of the image (*center favored scheme*, assuming the most important objects are always near the image center) or conversely to regions adjacent to the image boundary (*border favored scheme*, assuming images with similar semantics have similar backgrounds). Another choice is *area percentage scheme*. It uses the percentage of the image covered by a region as the weight for that region based on the viewpoint that important objects in an image tend to occupy larger areas.

In the UFM measure, both area percentage and border favored schemes are used. The weight vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ are defined as $\mathbf{w}_1 = (1-\lambda)\mathbf{w}_a + \lambda\mathbf{w}_b$ and $\mathbf{w}_2 = \mathbf{w}_a$

where $\mathbf{w}_a$ contains the normalized area percentages of the query and target images, $\mathbf{w}_b$ contains normalized weights [2] which favor regions near the image boundary, $\lambda \in [0,1]$ adjusts the significance of $\mathbf{w}_a$ and $\mathbf{w}_b$ in $\mathbf{w}_1$. The weights ρ_1 and ρ_2 are given by

$$\begin{aligned} \rho_1 &= 1-\rho, \\ \rho_2 &= \rho, \end{aligned}$$

where ρ is within the real interval $[0,1]$. Consequently, the UFM measure for query image q and target image t is defined as

$$m_{(q,t)} = (1-\rho)\left[(1-\lambda)\mathbf{w}_a + \lambda \mathbf{w}_b\right]^T \mathbf{L}^{(\mathcal{F}_q,\mathcal{F}_t)} + \rho \mathbf{w}_a^T \mathbf{L}^{(\mathcal{H}_q,\mathcal{H}_t)}. \tag{5.13}$$

As shown by equation (5.13), the UFM measure incorporates three similarity components captured by $\mathbf{w}_a^T \mathbf{L}^{(\mathcal{F}_q,\mathcal{F}_t)}$, $\mathbf{w}_b^T \mathbf{L}^{(\mathcal{F}_q,\mathcal{F}_t)}$, and $\mathbf{w}_a^T \mathbf{L}^{(\mathcal{H}_q,\mathcal{H}_t)}$:

- $\mathbf{w}_a^T \mathbf{L}^{(\mathcal{F}_q,\mathcal{F}_t)}$ contributes to the UFM measure from a color and texture perspective because $\mathbf{L}^{(\mathcal{F}_q,\mathcal{F}_t)}$ reflects the color and texture resemblance between the query and target images. In addition, the matching of regions with larger areas is favored which is the direct consequence of the area percentage weighting scheme.

- $\mathbf{w}_b^T \mathbf{L}^{(\mathcal{F}_q,\mathcal{F}_t)}$ also expresses the color and texture resemblance between images. But, unlike in $\mathbf{w}_a^T \mathbf{L}^{(\mathcal{F}_q,\mathcal{F}_t)}$, regions adjacent to the image boundaries are given a higher preference because of the border favored weight vector $\mathbf{w}_b$. Intuitively, $\mathbf{w}_b^T \mathbf{L}^{(\mathcal{F}_q,\mathcal{F}_t)}$ characterizes the similarity between the backgrounds of images.

[2] Both the summation of all entries of $\mathbf{w}_a$ and that of $\mathbf{w}_b$ equal 1.

- Similarly, $\mathbf{w}_a^T \mathbf{L}^{(\mathcal{H}_q,\mathcal{H}_t)}$ describes the similarity of the shape properties of the regions (or objects) in both images since $\mathbf{L}^{(\mathcal{H}_q,\mathcal{H}_t)}$ contains similarity measures for shape features.

Weighted by λ and ρ, the aforementioned similarity components are then synthesized into the UFM measure, in which $[(1-\lambda)\mathbf{w}_a + \lambda \mathbf{w}_b]^T \mathbf{L}^{(\mathcal{F}_q,\mathcal{F}_t)}$ represents the color and texture similarity with contributions from the area percentage and the border favored schemes weighted by λ, while ρ determines the significance of the shape similarity, $\mathbf{w}_a^T \mathbf{L}^{(\mathcal{H}_q,\mathcal{H}_t)}$, with respect to the color and texture similarity. In our system, the query image is automatically classified as either a textured or a non-textured image (for details see [65]). For textured images, the information of the shape similarity is skipped ($\rho = 0$) in the UFM measure since region shape is not perceptually important for such images. For non-textured images, ρ is chosen to be 0.1. Experiments indicate that including shape similarity as a small fraction of the UFM measure can improve the overall performance of the system. We intentionally stress color and texture similarities more than shape similarity because, compared with the color and texture features, shape features used in our system are more sensitive to image segmentation. The weight parameter λ is set to be 0.1 for all images. Experiments show that large λ is beneficial to categorizing images with similar background patterns. For example, the background of images of flowers often consists of green leaves and images of elephants are very likely to have trees in them. Thus emphasizing backgrounds can help grouping images, such as flowers or elephants, together. But the above background assumption is in general not true. In our observation, the overall image categorization performance degrades significantly for

$\lambda > 0.5$. When ρ and λ are within $[0.05, 0.3]$, no major system performance deterioration is noticed in our experiments.

$m_{(q,t)}$ is always in the real interval $[0, 1]$ because $\mathbf{w}_a$ and $\mathbf{w}_b$ are normalized weight vectors, and ρ and λ are within $[0, 1]$. It is easy to check that $m_{(q,t)} = 1$ if two images are same. The experiments show that there is little resemblance between images if $m_{(q,t)} \leq 0.5$. In this sense, the UFM measure is very intuitive for query users.

5.3.4 An Algorithmic View

An algorithmic outline of the UFM algorithm is given as below. Weights $\rho, \lambda \in [0, 1]$ are fixed. Inputs are $(\mathcal{F}_q, \mathcal{H}_q)$ (characterized by $\hat{\mathbf{f}}_j \in \mathbb{R}^6$, $d_f \in \mathbb{R}$, $\mathbf{h}_j \in \mathbb{R}^3$, $d_h \in \mathbb{R}$, $1 \leq j \leq C_q$), $(\mathcal{F}_t, \mathcal{H}_t)$ (characterized by $\hat{\mathbf{f}'}_j \in \mathbb{R}^6$, $d'_f \in \mathbb{R}$, $\mathbf{h'}_j \in \mathbb{R}^3$, $d'_h \in \mathbb{R}$, $1 \leq j \leq C_t$), and weight vectors $\mathbf{w}_a, \mathbf{w}_b \in \mathbb{R}^{C_q+C_t}$. The UFM measure $m_{(q,t)}$ is the output.

Algorithm 5.2. Unified Feature Matching

1 FOR $i = 1$ TO C_q

2 $\quad \mathbf{L}^{(\mathcal{F}_q,\mathcal{F}_t)}[i] \leftarrow \dfrac{d_f + d'_f}{d_f + d'_f + \min_{j=1,\ldots,C_t} \|\hat{\mathbf{f}}_i - \hat{\mathbf{f}'}_j\|}$

3 $\quad$ IF the query image is non-textured

4 $\quad\quad \mathbf{L}^{(\mathcal{H}_q,\mathcal{H}_t)}[i] \leftarrow \dfrac{d_h + d'_h}{d_h + d'_h + \min_{j=1,\ldots,C_t} \|\mathbf{h}_i - \mathbf{h'}_j\|}$

5 $\quad$ END

6 END

7 FOR $i = 1$ TO C_t

8 $\quad \mathbf{L}^{(\mathcal{F}_q,\mathcal{F}_t)}[i + C_q] \leftarrow \dfrac{d_f + d'_f}{d_f + d'_f + \min_{j=1,\ldots,C_q} \|\hat{\mathbf{f}'}_i - \hat{\mathbf{f}}_j\|}$

```
 9     IF the query image is non-textured
```

10 $\quad \mathbf{L}^{(\mathcal{H}_q,\mathcal{H}_t)}[i + C_q] \leftarrow \frac{d_h + d'_h}{d_h + d'_h + \min_{j=1,\ldots,C_q} \|\mathbf{h'}_i - \mathbf{h}_j\|}$

```
11     END
12 END
```

13 $m_{(q,t)} \leftarrow [(1-\lambda)\mathbf{w}_a + \lambda \mathbf{w}_b]^T \mathbf{L}^{(\mathcal{F}_q,\mathcal{F}_t)}$

```
14 IF the query image is non-textured
```

15 $\quad m_{(q,t)} \leftarrow (1-\rho) m_{(q,t)} + \rho \mathbf{w}_a^T \mathbf{L}^{(\mathcal{H}_q,\mathcal{H}_t)}$

```
16 END
```

5.4 An Algorithmic Summarization of the System

Based on the results given in Section 5.2 and Section 5.3, we describe the overall image retrieval and indexing scheme as follows.

1. Pre-processing image database

   To generate the codebook for an image database, signatures for all images in the database are extracted by Algorithm 5.1. Each image is classified as either a textured or a non-textured image using techniques in [65]. The whole process is very time-consuming. Fortunately, for a given image database, it is performed once for all.

2. Pre-processing query image

   Here we consider two scenarios, namely inside query and outside query. For inside query, the query image is in the database. Therefore, the fuzzy features and *semantic types* (textured or non-textured image) can be directly loaded from the

codebook. If a query image is not in the database (outside query), the image is first expanded or contracted so that the maximum value of the resulting width and height is 384 and the aspect ratio of the image is preserved. Fuzzy features are then computed for the resized query image. Finally, the query image is classified as textured or non-textured image.

3. Computing the UFM measures

   Using Algorithm 5.2, the UFM measures are evaluated for the query image and all images in the database, which have semantic types identical to that of the query image.

4. Returning query results

   Images in the database are sorted in a descending order according to the UFM measures obtained from the previous step. Depending on a user specified number n, the system returns the first n images. The *quick sort* algorithm is applied here.

Chapter 6

Cluster-Based Retrieval of Images by Unsupervised Learning

This chapter starts with some motivations for the cluster-based image retrieval method. Then we describe the general methodology of method in Section 6.2. Algorithmic and computational issues are discussed in Section 6.3. Section 6.4 introduces an image retrieval system using the proposed method.

6.1 Overview

Figure 6.1 shows a query image and the top 29 target images returned by a CBIR system described in [16] where the query image is on the upper-left corner. From left to right and top to bottom, the target images are ranked according to decreasing values of similarity measure. In essence, this can be viewed as a one-dimensional visualization of the image database in the "neighborhood" of the query image using a similarity measure. If the query image and majority of the images in the "vicinity" have the same semantics, then we would expect good results. But target images with high feature similarities to the query image may be quite different from the query image in terms of semantics due to the semantic gap. For the example in Figure 6.1, the target images belong to several semantic classes where the dominant ones include horses (11 out of 29), flowers (7 out of 29), golf player (4 out of 29), and vehicle (2 out of 29).

Fig. 6.1. A query image and its top 29 matches returned by the CBIR system at `http://wang.ist.psu.edu/IMAGE` (UFM). The query image is on the upper-left corner. The ID number of the query image is 6275.

However, the majority of top matches in Figure 6.1 belong to a quite small number of distinct semantic classes, which suggests a hypothesis that, in the "vicinity" of the query image, images of the same semantics are more similar to each other than to images of different semantics. Or, in other words, images tend to be semantically clustered. Therefore, a retrieval method, which is capable of capturing this structural relationship, may render semantically more meaningful results to the user than merely a list of images sorted by a similarity measure. Similar hypothesis has been well studied in document (or text) retrieval [3] where strong supporting evidence has been presented [45].

This motivates us to tackle the semantic gap problem from the perspective of unsupervised learning. In this thesis, we propose an algorithm, CLUster-based rEtrieval of images by unsupervised learning (CLUE), to retrieve image clusters instead of a set of ordered images: the query image and neighboring target images, which are selected

according to a similarity measure, are clustered by an unsupervised learning method and returned to the user. In this way, relations among retrieved images are taken into consideration through clustering and may provide extra information for ranking and presentation. CLUE has the following characteristics:

- It is a cluster-based image retrieval scheme that can be used as an alternative to retrieving a set of ordered images. The image clusters are obtained from an unsupervised learning process based on not only the feature similarity of images to the query, but also how images are similar to each other. In this sense, CLUE aims to capture the underlying concepts about how images of the same semantics are alike and present to the users semantic relevant *clues* as to where to navigate.

- It is a similarity-driven approach that can be built upon virtually any symmetric real-valued image similarity measure. Consequently, our approach could be combined with many other image retrieval schemes including the relevance feedback approach with dynamically updated models of similarity measure. Moreover, as shown in Section 8.2, it may also be used as a part of the interface for keyword-based image retrieval systems.

- It provides a dynamic and local visualization of the image database using a clustering technique. The clusters are created depending on which images are retrieved in response to the query. Consequently, the clusters have the potential to be closely adapted to characteristics of a query image. Moreover, by constraining the collection of retrieved images to the neighborhood of the query image, clusters generated by CLUE provides a local approximation of the semantic structure of the whole

image database. Although the overall semantic structure of the database could be very complex and extremely difficult to identify by a computer program, locally it may be well described by a simple approximation such as clusters. This is in contrast to current image database statistical classification methods [95, 112, 120], in which the semantic categories are derived for the whole database in a preprocessing stage, and therefore are global, static, and independent of the query.

- It employs a *pairwise representation* of images, which is independent of the specific representation of image features. A set of n images is represented by $\frac{n(n+1)}{2}$ pairwise similarities not as a collection of points in a certain normed feature space. This is crucial for nonmetric image similarity measures (many commonly used similarity measures are indeed nonmetric [51]), under which the images cannot be embedded in a normed vector space. Using pairwise distances for image retrieval is not a new idea. In [87], the authors propose the use of the MDS technique to embed a group of images in a two-dimensional Euclidean space so that the pairwise distances are preserved as much as possible. However, their method requires that the images are mapped to distributions in a geometric color space to make "the 'axes of variation' be perceptually clear to the user" [87]. While our approach does not impose such a strict constraint on the image features. Furthermore, our approach provides a local "semantics summarization" of the image database using a clustering technique instead of projecting images onto a plane.

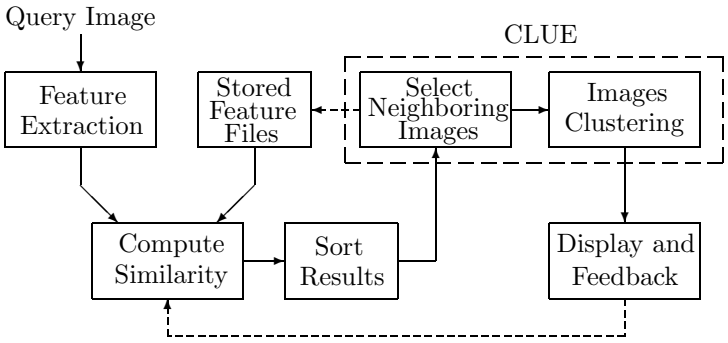


Fig. 6.2. A diagram of a general CBICR system. The arrows with dotted lines may not exist for some CBICR systems.

6.2 Retrieval of Similarity-Induced Image Clusters

For the purpose of simplifying the explanations, we call a CBIR system using CLUE a Content-Based Image Clusters Retrieval (CBICR) system. In this section, we first present an overview of CBICR systems. We then describe in detail the major components of CLUE, namely, neighboring image selection and images clustering.

6.2.1 System Overview

From a data-flow viewpoint, a general CBICR system can be characterized by the diagram in Figure 6.2. The retrieval process starts with feature extraction for a query image. The features for target images (images in the database) are usually precomputed and stored as feature files. Using these features together with an image similarity measure, the resemblance between the query image and target images are evaluated and sorted. Next, a collection of target images that are "close" to the query image are selected

as the neighborhood of the query image. A clustering algorithm is then applied to these target images. Finally, the system displays the image clusters and adjusts the model of similarity measure according to user feedback (if relevance feedback is included).

The major difference between CBICR and CBIR systems lies in the two processing stages, selecting neighboring target images and image clustering, which are the major components of CLUE. A typical CBIR system bypasses these two stages and directly outputs the sorted results to the display and feedback stage. Figure 6.2 suggests that CLUE can be designed independent of the rest of the components because the only information needed by CLUE is the sorted similarities. This implies that CLUE may be embedded in a typical CBIR system regardless of the image features being used, the sorting method, and whether there is feedback or not. The only requirement is a real-valued similarity measure satisfying the symmetry property. As a result, in the following subsections, we focus on the discussion of general methodology of CLUE, and assume that a similarity measure is given. An introduction of a specific CBICR system, which we have implemented, will be given in Section 6.4.

6.2.2 Neighboring Target Images Selection

To mathematically define the neighborhood of a point, we need to first choose a measure of distance. As to images, the distance can be defined by either a similarity measure (a larger value indicates a smaller distance) or a dissimilarity measure (a smaller value indicates a smaller distance). Because simple algebraic operations can convert a similarity measure into a dissimilarity measure, without loss of generality, we assume that the distance between two images is determined by a symmetric dissimilarity measure,

$d(i,j) = d(j,i) \geq 0$, and name $d(i,j)$ the distance between images i and j to simplify the notation.

Next we propose two simple methods to select a collection of neighboring target images for a query image i:

1. *Fixed radius method* (FRM) takes all target images within some fixed radius ϵ with respect to i. For a given query image, the number of neighboring target images is determined by ϵ.

2. *Nearest neighbors method* (NNM) first chooses k nearest neighbors of i as seeds. The r nearest neighbors for each seed are then found. Finally, the neighboring target images are selected to be all the distinct target images among seeds and their r nearest neighbors, i.e., distinct target images in $k(r+1)$ target images. Thus the number of neighboring target images is bounded above by $k(r+1)$.

If the distance is metric, both methods would generate similar results under proper parameters (ϵ, k, and r). However, for nonmetric distances, especially when the triangle inequality is not satisfied, the set of target images selected by two methods could be quite different regardless of the parameters. This is due to the violation of the triangle inequality: the distance between two images could be huge even if both of them are very close to a query image. Compared with the FRM, our empirical results show that, with proper choices of k and r, NNM tends to generate more structured collection of target images under a nonmetric distance.

On the other hand, the computational cost of NNM is higher than that of the FRM because of the extra time to find nearest neighbors for all k seeds. Thus a straightforward

implementation of NNM would be k-times slower than the FRM. Note that all k seeds are images in the database. Consequently, their nearest neighbors can be found in a preprocessing step to reduce the computational cost. However, the price we then have to pay is additional storage space for the nearest neighbors of target images. This work chooses NNM. A detailed discussion of computational issues (including parameters selection) will be covered in Section 6.3.

6.2.3 Spectral Graph Partitioning

Data representation is typically the first step to solve any clustering problem. Two types of representations are widely used: geometric representation and graph representation. When working with images, geometric representation has a major limitation: it requires that the images be mapped to points in some real normed vector space. Overall, this is a very restrictive constraint. For example, in region-based algorithms [16, 64, 120], an image is often viewed as a collection of regions. The number of regions may vary for different images. Although regions can be mapped to certain real normed vector space, it is in general impossible to do so for images in a lossless way unless the distance between images is metric, in which case embedding becomes feasible. Nevertheless, many distances defined for images are nonmetric for reasons given in [51].

Therefore, this thesis adopts a graph representation of neighboring target images. A set of n images is represented by a weighted undirected graph $G = (\mathbf{V}, \mathbf{E})$: the nodes $\mathbf{V} = \{1, 2, \ldots, n\}$ represent images, the edges $\mathbf{E} = \{(i, j) : i, j \in \mathbf{V}\}$ are formed between every pair of nodes, and the nonnegative weight w_{ij} of an edge (i, j), indicating the similarity between two nodes, is a function of the distance (or similarity) between nodes

(images) i and j. Given a distance $d(i,j)$ between images i and j, we define

$$w_{ij} = e^{-\frac{d(i,j)^2}{s^2}} \tag{6.1}$$

where s is a scaling parameter that needs to be tuned to get a suitable locality. The weights can be organized into a matrix $\mathbf{W}$, named the *affinity matrix*, with the ij-th entry given by w_{ij}. Although Equation (6.1) is a relatively simple weighting scheme, our experimental results (Section 8.2) have shown its effectiveness. The same scheme has been used in [34, 96, 123]. Support for exponential decay from psychological studies is provided by [34].

Under a graph representation, clustering becomes a graph partitioning problem. The Ncut described in Chapter 3.3 is recursively applied to get more than two clusters. But this leads to the questions: 1) which subgraph should be divided? and 2) when should the process stop? In this paper, we use a simple heuristic. The subgraph with the maximum number of nodes is recursively partitioned (random selection is used for tie breaking). The process terminates when the bound on the number of clusters is reached or the Ncut value exceeds some threshold.

6.2.4 Finding a Representative Image for a Cluster

Ultimately, the system needs to present the clustered target images to the user. Unlike a typical CBIR system, which displays certain numbers of top matched target images to the user, a CBICR system should be able to provide an intuitive visualization of the clustered structure in addition to all the retrieved target images. For this reason,

we propose a two-level display scheme. At the first level, the system shows a collection of representative images of all the clusters (one for each cluster). At the second level, the system displays all target images within the cluster specified by a user.

Nonetheless two questions still remain: 1) how to organize these clusters? and 2) how to find a representative image for each cluster? The organization of clusters will be described in Section 6.3.2. For the second question, we define a representative image of a cluster to be the image that is most similar to all images in the cluster. This statement can be mathematically illustrated as follows. Given a graph representation of images $G = (\mathbf{V}, \mathbf{E})$ with affinity matrix $\mathbf{W}$, let the collection of image clusters be $\{\mathbf{C}_1, \mathbf{C}_2, \cdots, \mathbf{C}_m\}$, which is also a *partition* of $\mathbf{V}$, i.e., $\mathbf{C}_i \cap \mathbf{C}_j = \emptyset$ for $i \neq j$ and $\bigcup_{i=1}^{m} \mathbf{C}_i = \mathbf{V}$. Then the representative node (image) of $\mathbf{C}_i$ is

$$\arg \max_{j \in \mathbf{C}_i} \sum_{t \in \mathbf{C}_i} w_{jt} \, . \tag{6.2}$$

Basically, for each cluster, we pick the image that has the maximum sum of within cluster similarities.

6.3 An Algorithmic View

This section starts with an algorithmic summary of CLUE described in Section 6.2. We then talk about the organization of clusters, followed by a discussion of computational complexity and parameters selection.

6.3.1 Outline of Algorithm

The following pseudo code selects a group of neighboring target images for a query image, recursively partitions the query image and target images using the Ncut method, and outputs the clusters together with their representative images.

Algorithm 6.1. CLUE

Inputs: A query image; $k \geq 2$ and $r \geq 1$ needed by NNM for neighboring target images selection; $M \geq 2$ (maximum number of clusters) and $0 \leq T \leq 1$ (threshold for the Ncut value) required by the recursive Ncut method.

Outputs: Image clusters and the corresponding representative images.

[Generating neighboring target images]

```
get k nearest neighbors (seeds) of the query image and denote the
results as {S_10, S_20, ..., S_k0}
let I be an empty set
FOR i = 1 TO k
    get r nearest neighbors of seed S_i0 and denote the results as
    {S_i1, S_i2, ..., S_ir}
    FOR j = 0 TO r
        IF S_ij ∉ I
            I ← I ∪ {S_ij}
        END
    END
END
```

[Graph construction]

11 for the query image and all target images in $\mathbf{I}$, generate a weighted graph $G = (\mathbf{V}, \mathbf{E})$ with affinity matrix $\mathbf{W}$

[Recursive Ncut]

12 $m \leftarrow 1$

13 $v \leftarrow 1$

14 $\mathcal{C} \leftarrow \{\mathbf{V}\}$

15 WHILE ($m < M$ AND $v < T$)

16 $\quad \mathbf{P} \leftarrow \arg\max_{\mathbf{C} \in \mathcal{C}} |\mathbf{C}|$ ($|\mathbf{C}|$ denotes the volume of $\mathbf{C}$)

17 $\quad$ use the Ncut algorithm to partition $\mathbf{P}$ into two disjoint sets $\mathbf{A}$ and $\mathbf{B}$

18 $\quad v \leftarrow Ncut(\mathbf{A}, \mathbf{B})$

19 $\quad \mathcal{C} \leftarrow (\mathcal{C} - \{\mathbf{P}\}) \cup \{\mathbf{A}, \mathbf{B}\}$

20 $\quad m \leftarrow m + 1$

21 END

22 FOR each element in $\mathcal{C}$

23 $\quad$ find the representative image according to (6.2)

24 END

25 OUTPUT image clusters and the corresponding representative images

In the above pseudo code, lines $1 - 10$ generate the neighboring target images for a query image using NNM. Line 11 constructs a weighted undirected graph for the query image and its neighboring target images. Lines $12 - 21$ apply the Ncut algorithm

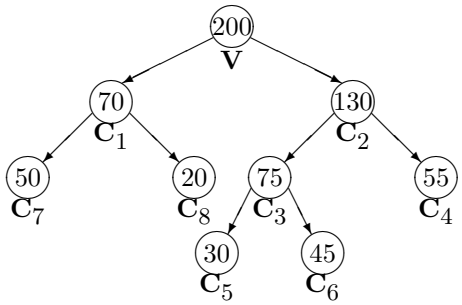


Fig. 6.3. A tree generated by four Ncuts that are applied to **V** with 200 nodes. The numbers denote the size of the corresponding clusters.

recursively to the graph or the largest subgraph until a bound on the number of clusters is reached or the Ncut value exceeds a predefined threshold. The number of clusters then equals m. The representative images for the clusters are found in lines $22 - 24$.

6.3.2 Organization of Clusters

The recursive Ncut partition described by lines $12 - 21$ of the pseudo code is essentially a hierarchical divisive clustering process that produces a tree. For example, Figure 6.3 shows a tree generated by four recursive Ncuts. The first Ncut divides **V** into $\mathbf{C}_1$ and $\mathbf{C}_2$. Since $\mathbf{C}_2$ has more nodes than $\mathbf{C}_1$, the second Ncut partitions $\mathbf{C}_2$ into $\mathbf{C}_3$ and $\mathbf{C}_4$. Next, $\mathbf{C}_3$ is further divided because it is larger than $\mathbf{C}_1$ and $\mathbf{C}_4$. The fourth Ncut is applied to $\mathbf{C}_1$, and gives the final five clusters (or leaves): $\mathbf{C}_4$, $\mathbf{C}_5$, $\mathbf{C}_6$, $\mathbf{C}_7$, and $\mathbf{C}_8$.

The above example suggests trees as a natural organization of clusters, which could be presented to the user. Nonetheless, the tree organization here may be misleading

to a user because there is no guarantee of any correspondence between the tree and the semantic structure of images. Furthermore, organizing image clusters into a tree structure will significantly complicate the user interface. So, in this work, we employ a simple linear organization of clusters called *traversal ordering*: arrange the leaves in the order of a binary tree traversal (left child goes first). For the example in Figure 6.3, it yields a sequence: $\mathbf{C}_7$, $\mathbf{C}_8$, $\mathbf{C}_5$, $\mathbf{C}_6$, and $\mathbf{C}_4$. However, the order of two clusters produced by an Ncut bipartition iteration is still undecided, i.e., which one should be the *left child* and which one should be the *right child*. This can be solved by enforcing an arbitration rule: 1) let $\mathbf{C}_1$ and $\mathbf{C}_2$ be two clusters generated by an Ncut on $\mathbf{C}$, and d_1 (d_2) be the minimal distance between the query image and all images in $\mathbf{C}_1$ ($\mathbf{C}_2$); 2) if $d_1 < d_2$ then $\mathbf{C}_1$ is the left child of $\mathbf{C}$, otherwise, $\mathbf{C}_2$ is the left child.

The traversal ordering and arbitration rule have the following properties:

- The query image is in the leftmost leaf ($\mathbf{C}_7$ in Figure 6.3) since a cluster containing the query image will have a minimum distance (d_1 or d_2) of 0, and thus will always be assigned to the left child. (Note that $\mathbf{V}$ includes the query image).

- We can view d_1 (or d_2) as a distance from a query image to a cluster of images. In this sense, for any parent node, its left child is closer to the query image than its right child.

- In the traversal, the leaves of the left subtree of any parent node appear before the leaves of its right subtree.

Therefore, the resulting linear organization of clusters considers not only the distances to a query image, but also the hierarchical structure that generates the clusters. To this

end, it may be viewed as a structured sorting of clusters in ascending order of distances to a query image. For the sake of consistency, images within each cluster are also organized in ascending order of distances to a query image.

6.3.3 Computational Complexity

The computational complexity of a CBICR system is higher than that of a typical CBIR system due to the added computation of CLUE. The time complexity of CLUE is the sum of the complexity of NNM and the complexity of the recursive Ncut.

Since NNM needs to find r nearest neighbors for all k seeds, a straightforward implementation, which treats each seed as a new query, would make the whole process very slow when the size of image database is large. For example, using a 700MHz Pentium III PC, the SIMPLIcity [120] system with UFM (Unified Feature Matching) [16] similarity measure, on average, takes 0.7 second to index a query image (time for computing and sorting the similarities between the query image and all target images, excluding the time for feature extraction) on a database of $60,000$ images [1]. It adds up to 21 seconds for NNM if $k = 30$, i.e., 30 seeds are used and each seed takes 0.7 second on average. This is certainly an excessive amount of time for a real-time retrieval system.

Two methods can be applied to reduce the time cost of NNM. One method is to parallelize NNM because nearest neighbors for all k seeds can be selected simultaneously. The other method utilizes the fact that all seeds are images in the database. Thus similarities can be computed and sorted in advance. So the time needed by NNM does

[1] The time complexity is $O(C^2N + NlogN)$ where N is the size of the database, C is the average number of regions of an image [16].

not scale up by the number of seeds. Nevertheless, it then requires storing the sorting results with every image in the database as a query image. The space complexity becomes $O(N^2)$ where N is the size of the database. However, the space complexity can also be reduced because NNM only needs r nearest neighbors, which leads to a space complexity of $O(rN)$. The locality constraint guarantees that r is very small compared with N. In our implementation, only the ID numbers of the 100 nearest neighbors for each image are stored ($N = 60,000$). The second method is used in our experimental system. We argue that this method is practical even if the database is very large. Because computing and sorting similarities for all target images may be very time-consuming, this process is required only once. Moreover, the process can also be parallelized for each target image. If new images are added to the database, instead of redoing the whole process, we can merely compute those similarities associated with new images and update previously stored sorting results accordingly.

The time needed by the recursive Ncut process consists of two parts: graph construction and the Ncut algorithm. For graph construction, one needs to evaluate $\frac{n(n+1)}{2}$ entries of the affinity matrix where $n \leq k(r+1)+1$ is the number of nodes (query image and all its neighboring target images). Thus the time complexity is $O(n^2)$. The Ncut algorithm involves eigenvector computations, of which the time complexity is $O(n^3)$ using standard eigensolvers. Fortunately, we only need to compute the second smallest generalized eigenvector, which can be solved by the Lanczos algorithm (Ch.9, [39]) in $O(n^2)$. Note that if the affinity matrix is sparse, the time complexity of the Lanczos algorithm is $O(n)$. Yet in our application, the sparsity is in general not guaranteed. As

the number of clusters is bounded by M, the total time complexity for the recursive Ncut process is $O(k^2r^2)$ (because $n \leq k(r+1)+1$).

6.3.4 Parameters Selection

Several parameters need to be specified to implement Algorithm 6.1. These include k and r for NNM, s for affinity matrix evaluation, M and T for recursive Ncut. Three requirements are considered when specifying k and r. First, we want the neighboring images to be close to the query image so that the assumption of a locally clustered structure is valid. Second, we need sufficient number of images to provide an informative local visualization of the image database to the user. Third, computational cost should be kept within the tolerance of real-time applications. It is clear that the second constraint favors large k and r, while the other two constraints need k and r to be small. Finding a proper tradeoff is dependent upon the application.

For the CBICR system described in the next section, k and r are obtained from a simple tuning strategy. We randomly choose 20 query images from the image database. For each pair of k and r, where $k \in \{25, 26, \cdots, 35\}$ and $r \in \{5, 6, \cdots, 10\}$, we manually examine the semantics of images generated by NNM using each of the 20 query images, and record the average number of distinct semantics. Next, all pairs of k and r corresponding to the median of the above recorded numbers are found. We pick the pair with minimal kr value, which gives $k = 29$ and $r = 7$ for our system. As a byproduct, M (maximum number of clusters) in recursive Ncut is set to be 8, which is the integer closest to the median. Note that our criteria on distinct semantics may be very different

from the criteria of a system user. However, we observed that the system is not sensitive to k and r. This will be demonstrated numerically in Section 8.2.4.

The parameter s in (6.1) reflects the local scale on distances. Thus it should be adaptive to the query image and its neighboring target images. In our system, $s = 2\sigma$ where σ is the standard deviation of all the pairwise distances used to construct the affinity matrix. The threshold T is chosen to make the median of the number of clusters generated by recursive Ncuts on the 20 collections of images, which are used in k and r tuning process, equal or close to $M = 8$. A proper T value is found to be 0.9.

6.4 A Content-Based Image Clusters Retrieval System

Our CBICR system uses the same feature extraction scheme and similarity measure (UFM) as those described in Chapter 5. It has a very simple CGI-based query interface. The system provides a *Random* option that will give a user a random set of images from the image database to start with. In addition, users can either enter the ID of an image as the query or submit any image on the Internet as a query by entering the URL of the image. The system is capable of handling any standard image format from anywhere on the Internet and reachable by our server via the HTTP protocol.

Once a query image is received, the system displays a list of thumbnails each of which represents an image cluster. The thumbnails are found according to (6.2), and sorted using the algorithm in Section 6.3.2. Figure 6.4(a) shows 8 clusters corresponding to a query image with ID 6275. Below each thumbnail are cluster ID and the number of images in that cluster. A user can start a new query search by submitting a new image ID or URL, get a random set of images from the image database, or click a thumbnail

(a) Thumbnails of image clusters.

(b) Images in Cluster 1.

Fig. 6.4. Two snapshots of the user interface displaying query results for a query image with ID 6275.

to see all images in the associated cluster. The contents of Cluster 1 are displayed in Figure 6.4(b). From left to right and top to bottom, the images are listed in ascending order of distances to the query image. The underlined numbers below the images are image IDs. The other numbers are cluster IDs. The image with a border around it is the representative image for the cluster. Again, a user has three options: enter a new image ID or URL, get a random set of images from the database, or click an image to submit it as a query.

Chapter 7

Image Categorization by Learning and Reasoning with Regions

This chapter starts with a motivational discussion for region-based image categorization. A detailed description of a new method is presented in Section 7.2 and 7.3.

7.1 Overview

Although color and texture are fundamental aspects for visual perception, human discernment of certain visual contents could be potentially associated with interesting classes of objects or semantic meaning of objects in the image. For one example: if we are asked to decide which images in Figure 7.1 are images about *winter*, *people*, *skiing*, and *outdoor scenes*, at a single glance, we may come up with the following answers together with supporting arguments:

- Images (a) to (d) are winter images since we see *snow* in them;
- Images (b) to (f) are images about people since there are *people* in them;
- Images (b) to (d) are images about skiing since we see *people and snow*;
- All images listed in Figure 7.1 are outdoor scenes since they all have a region or regions corresponding to *snow*, *sky*, *sea*, *trees*, *or grass*.

(a) (b) (c) (d) (e) (f) (g)

Fig. 7.1. Sample images belonging to at least one of the categories: winter, people, skiing, and outdoor scenes.

This seems to be effortless for humans because prior knowledge of similar images and objects may provide powerful assistance for us in recognition. Given a set of labeled images, can a computer program learn such knowledge or semantic concepts from implicit information of objects contained in images? In this work, we propose an image categorization method using a set of automatically extracted rules. Intuitively, these rules bear an analogy to the supporting arguments that are used to describe a semantic concept about images in the above example.

In terms of image representation, our approach is a region-based method. Images are segmented into regions such that each region is roughly homogeneous in color and texture. Each region is characterized by one feature vector describing color, texture, and shape attributes. Consequently, an image is represented by a collection of feature vectors. If segmentation is ideal, regions will correspond to objects. But as we have mentioned earlier, semantically accurate image segmentation by a computer program is still an ambitious long-term goal for computer vision researchers. Nevertheless, we argue that region-based image representation can provide some useful information about objects even though segmentation may not be perfect. Moreover, empirical results in

Section 8.3 demonstrate that the proposed method is not sensitive to inaccurate image segmentation.

From the perspective of learning or classifier design, our approach can be viewed as a generalization of supervised learning in which labels are associated with images instead of individual regions. This is in essence identical to MIL setting [27, 68, 131] where images and regions are respectively called *bags* and *instances* [1]. While every instance may possess an associated true label, it is assumed that instance labels are only indirectly accessible through labels attached to bags. Several researchers have applied MIL for image classification and retrieval [2, 67, 130]. Key assumptions of their formulation of MIL are that *bags and instances share the same set of labels (or categories or classes or topics); and a bag receives a particular label if at least one of the instances in the bag possesses the label.* For binary classification, this implies that a bag is "positive" if at least one of its instances is a positive example; otherwise, the bag is "negative." Therefore, learning focuses on finding which of the instances in a positive bag are the actual positive examples and which ones are not.

However, this formulation of MIL does not perform well for image categorization even if image segmentation and object recognition are assumed to be ideal. For one simple example, let's consider the sample images in Figure 7.1 with *skiing* being the positive class. It should be clear that images (b), (c), and (d) are positive images; images (a), (e), (f), and (g) are negative images. In this example, any object in a positive image also appears in at least one of the negative images: *snow* appears in (a); *people* and *sky* appear in (e) and (f); *trees* appears in (a), (f), and (g). Hence, to correctly classify

[1] In this chapter, the terms bag (instance) and image (region) have identical meaning.

positive images, some of these objects need positive labels. But labeling any of these objects positive (note that labels for the same object will be consistent across images) will inevitably misclassify some negative images. Although using the co-occurrence of *snow* and *people* will avoid the paradox, it is not allowed by the above formulation of MIL. Inaccurate segmentation and recognition will only worsen the situation.

This motivates our approach under much weaker assumptions:

1. Bags and instances do not share the same set of labels (or categories or classes or topics). Only the set of bag labels, not the set of instance labels, is given in advance. For example, {*winter*, *people*, *skiing*, *outdoor scenes*} is the set of bag (or image) labels for images in Figure 7.1. While a somewhat ideal (but unknown) set of instance (or region) labels would be descriptions of instance semantic categories in all the bags: {*snow*, *people*, *sky*, *sea*, *trees*, *grass*}.

2. Each instance has multiple labels with different weights. The weight, named *degree of membership*, illustrates the degree of wellness with which a corresponding instance label characterizes the instance, thus, to a certain extent, models the uncertainties associated with image segmentation. For instance, an under-segmented region may contain both *trees* and *grass*; an over-segmented *sky* may look similar to both *sky* and *sea*.

3. The label of a bag is determined collectively by degrees of membership of its instances with respect to all instance labels.

Our approach proceeds as follows. First, in the space of region features, a collection of feature vectors, each of which is called a region prototype (RP), is determined

according to an objective function, Diverse Density (DD) [68], defined over the region feature space. DD measures a co-occurrence of similar regions from *different* images in the *same* category. Each RP is chosen to be a local maximizer of DD. Hence, loosely speaking, an RP represents a class of regions that is more likely to appear in images with the specific label than in the other images. In the context of our first assumption above, each RP corresponds to an instance class. Next, an image classifier is defined by a set of rules associating the appearance of RPs in an image (described by degrees of membership of regions with respect to the RPs) with image labels. We formulate the learning of such classifiers as an SVM problem [9, 115]. Consequently, a collection of SVMs are trained, each corresponding to one image category.

7.2 Learning Region Prototypes Using Diverse Density

In this section, we first present the basic concepts of Diverse Density (DD), which is proposed by Maron and Lozano-Pérez [68] for learning from multiple-instance examples. We then introduce a scheme to extract region prototypes using DD.

7.2.1 Diverse Density

We start with some notations in MIL. Let $\mathcal{D}$ be the labeled data set which consists of l bag/label pairs, i.e., $\mathcal{D} = \{(\mathbf{B}_1, Y_1), \cdots, (\mathbf{B}_l, Y_l)\}$. Each bag $\mathbf{B}_i$ is a collection of instances with $\mathbf{x}_{ij}$ denoting the jth instance in the bag. Different bags may have different number of instances. Labels Y_i take binary values 1 or -1. A bag is called a positive bag if its label is 1; otherwise, negative bag. Note that a label is attached to each bag and not to every instance. In the context of images, a bag is a collection of region feature vectors;

an instance is a region feature vector (as defined in Chapter 5.2); positive (negative) label represents that an image belongs (does not belong) to a particular category.

Given a set of labeled bags, finding what is in common among the positive bags and does not appear in the negative bags may provide inductive clues for classifier design. In the ideal scenario, these clues can be extracted by the intersection of the positive bags minus the union of the negative bags. However, in practice strict set operations of intersection, union, and difference may not be useful because most real world problems involve noisy information: features of instances might be corrupted by noise; some labels of bags might be wrong; strict intersection of positive bags might generate an empty set. DD implements soft versions of the intersection, union, and difference operations by thinking of the instances and bags as generated by some probability distribution. It is a function defined over the instance feature space. The DD value at a point in the feature space is indicative of the probability that the point agrees with the underlying distribution of positive and negative bags.

Next, we introduce one definition of DD from [68]. Interested readers are refereed to [68] for detailed derivations based on a probabilistic framework. Given a labeled data set $\mathcal{D}$, the DD function is defined as

$$DD_{\mathcal{D}}(\mathbf{x}, \mathbf{w}) = \prod_{i=1}^{l} \left[\frac{1+Y_i}{2} - Y_i \prod_{j=1}^{C_i} \left(1 - e^{-\|\mathbf{x}_{ij} - \mathbf{x}\|^2_{\mathbf{w}}} \right) \right] . \tag{7.1}$$

Here, $\mathbf{x}$ is a point in the feature space of instances; $\mathbf{w}$ is a weight vector defining which features are considered important and which are considered unimportant; C_i is the number of instances in the ith bag; and $\|\cdot\|_{\mathbf{w}}$ denotes a weighted norm defined by

$$\|\mathbf{x}\|_{\mathbf{w}} = \left[\mathbf{x}^T \mathrm{Diag}(\mathbf{w})^2 \mathbf{x}\right]^{\frac{1}{2}} \tag{7.2}$$

where $\mathrm{Diag}(\mathbf{w})$ is a diagonal matrix whose (i,i) the entry is the ith component of $\mathbf{w}$.

It is clear that values of DD are always between 0 and 1. For fixed $\mathbf{w}$, if a point $\mathbf{x}$ is close to an instance from a positive bag $\mathbf{B}_i$, then $\frac{1+Y_i}{2} - Y_i \prod_{j=1}^{C_i} \left(1 - e^{-\|\mathbf{x}_{ij}-\mathbf{x}\|^2_{\mathbf{w}}}\right)$ will be close to 1; if $\mathbf{x}$ is close to an instance form a negative bag $\mathbf{B}_i$, then $\frac{1+Y_i}{2} - Y_i \prod_{j=1}^{C_i} \left(1 - e^{-\|\mathbf{x}_{ij}-\mathbf{x}\|^2_{\mathbf{w}}}\right)$ will be close to 0. The above definition indicates that $DD(\mathbf{x}, \mathbf{w})$ will be close to 1 if $\mathbf{x}$ is close to instances from different positive bags and, at the same time, far away from instances in all negative bags. Thus it measures a co-occurrence of instances from different (diverse) positive bags.

7.2.2 Learning Region Prototypes

For the applications discussed in this thesis, the DD function defined in (7.1) is a continuous and highly nonlinear function with multiple peaks and valleys (or local maximums and minimums). A larger value of DD at a point indicates a higher probability that the point fits more with the instances from positive bags than with those from negative bags. This motivates us to choose local maximizers of DD as region prototypes (RPs). Loosely speaking, an RP represents a class of regions that is more likely to appear in positive bags than in negative bags. For the sample images in Figure 7.1, if *winter*

category is chosen to be the positive class, one may expect to find an RP corresponding to regions of *snow* because, in this example, every winter image ((a), (b), (c), and (d)) contains a region or regions of *snow*; and *snow* does not show up in the rest images ((e), (f), and (g)).

Therefore learning RPs becomes an optimization problem: finding local maximizers of the DD function in a high-dimensional space. Since the DD functions are smooth, we apply gradient based methods to find local maximizers. Now the question is: how do we find all the local maximizers? In fact we do not know in general the number of local maximizers a DD function has. However, according to the definition of DD, a local maximizer is close to instances from positive bags [68]. Thus starting a gradient based optimization from one of those instances will likely lead to a local maximum. Therefore, a simple heuristic is applied to search for multiple maximizers: we start an optimization at every instance in every positive bag with uniform weights, and record all the resulting maximizers (feature vector and corresponding weights).

RPs are selected from those maximizers satisfying two additional constraints: 1) they need to be distinct from each other; and 2) they need to have large DD values. The first constraint concerns with the precision issue of numerical optimization: due to numerical precision, different starting points may lead to different versions of the same maximizer. So we need to remove some of the maximizers that are essentially repetitions of each other. The second constraint limits RPs to those that are most informative in terms of co-occurrence in different positive bags. In our algorithm, this is achieved by picking maximizers with DD values greater than certain threshold.

According to the above steps, one can find RPs representing classes of regions that are more likely to appear in positive bags than in negative bags. One could argue that RPs with an exactly reversed property (more likely to appear in negative bags than in positive bags) may be of equal importance. Such RPs can be computed in exactly the same steps after switching the labels of positive and negative bags. Our empirical study shows that including such RPs (for negative bags) can improve classification accuracy.

7.2.3 An Algorithmic View

Next, we summarize the above discussion into pseudo code. The input is a set of labeled bags $\mathcal{D}$. The following pseudo code learns a collection of RPs each of which is represented as a pair of vectors $(\mathbf{x}_i^*, \mathbf{w}_i^*)$. The optimization problem involved is solved by Quasi-Newton search `dfpmin` in [84].

Algorithm 7.1. Learning RPs

MainLearnRPs($\mathcal{D}$)

```
1 R_p = LearnRPs(D)                          [Learn RPs for positive bags]
2 negate labels of all bags in D
3 R_n = LearnRPs(D)                          [Learn RPs for negative bags]
4 OUTPUT (the set union of R_p and R_n)
```

LearnRPs($\mathcal{D}$)

```
1 set P be the set of instances from all positive bags in D
2 initialize M to be an empty set
```

3 FOR (every instance in $\mathbf{P}$ as starting point for $\mathbf{x}$)

4 set the starting point for $\mathbf{w}$ to be all 1's

5 find a maximizer $(\mathbf{p}, \mathbf{q})$ of the $\log(DD)$ function by quasi-newton search

6 add $(\mathbf{p}, \mathbf{q})$ to $\mathbf{M}$

7 END

8 set $i = 1$, $T = \frac{\max_{(\mathbf{p},\mathbf{q})\in\mathbf{M}} log(DD_{\mathcal{D}}(\mathbf{p},\mathbf{q}))+\min_{(\mathbf{p},\mathbf{q})\in\mathbf{M}} log(DD_{\mathcal{D}}(\mathbf{p},\mathbf{q}))}{2}$

9 REPEAT

10 set $(\mathbf{x}_i^*, \mathbf{w}_i^*) = \arg\max_{(\mathbf{p},\mathbf{q})\in\mathbf{M}} \log(DD_{\mathcal{D}}(\mathbf{p}, \mathbf{q}))$

11 remove from $\mathbf{M}$ all elements $(\mathbf{p}, \mathbf{q})$ satisfying

$(\|\mathbf{p} - \mathbf{x}_i^*\| < \alpha\|\mathbf{x}_i^*\|$ AND $\|\text{abs}(\mathbf{q}) - \text{abs}(\mathbf{w}_i^*)\| < \alpha\|\mathbf{w}_i^*\|)$ OR $\log(DD_{\mathcal{D}}(\mathbf{p}, \mathbf{q})) < T$

12 set $i = i + 1$

13 WHILE ($\mathbf{M}$ is not empty)

14 OUTPUT $(\{(\mathbf{x}_1^*, \mathbf{w}_1^*), \cdots, (\mathbf{x}_{i-1}^*, \mathbf{w}_{i-1}^*)\})$

In the above pseudo code for **LearnRPs**, lines 1–7 find a collection of local maximizers for the DD function by starting optimization at every instance in every positive bag with uniform weights. For a better numerical stability, the optimization is performed on the $\log(DD)$ function, in stead of the DD function itself. Lines 8–13 describe an iterative process to pick a collection of "distinct" local maximizers as RPs. In each iteration, an element of $\mathbf{M}$, which is a local maximizer, with the maximal $\log(DD)$ value (or, equivalently, the DD value) is selected as an RP (line 10). Then depending on the distances to the RP selected in this iteration and the DD values, elements, which are close the RP or have DD values lower than a threshold, are removed from $\mathbf{M}$ (line 11). A

new iteration starts if $\mathbf{M}$ is not empty. The abs($\mathbf{w}$) in line 11 computes component-wise absolute values of $\mathbf{w}$. This is because the signs in $\mathbf{w}$ has no effect on the definition (7.2) of weighted norm.

The number of RPs selected from $\mathbf{M}$ is determined by two parameters α and T. In our implementation, α is set to be 0.05; and T is the average of the maximal and minimal DD values for all local maximizers found (line 8). These two parameters may need to be adjusted for other applications. However, empirical study shows that the performance of the classifier, which will be discussed in the next section, is not sensitive to α and T.

7.3 Image Categorization by Reasoning with Region Prototypes

In this section, we present in details the modeling process which learns image classifiers based on RPs. We show that image categorization using regions can be naturally formulated as a rule-based classification problem. And under quite general assumptions, such classifiers are functionally equivalent to SVMs with kernels of certain forms. Therefore, SVM learning is applied to design the classifiers.

7.3.1 A Rule-Based Image Classifier

Prior knowledge of similar images and objects may be crucial for human to identify semantic meanings of images. As indicated by the simple example in Section 7.1, a human being can easily classify images into different categories by reasoning on the semantic meanings of objects in the images. In that specific problem setting, the class membership of an image can be described by a set of simple rules of the form:

- If there is *snow* in an image, then the image is about *winter*;
- If there is *people* in an image, then the image is about *people*;
- If there are *snow* AND *people* in an image, then the image is about *skiing*;
- If there are *snow* OR *sky* OR *sea* OR *trees* OR *grass* in an image, then the image is about *outdoor scene*.

This motivates us to classify images using a set of rules describing whether of not some RPs appear in an image.

How can one decide if an RP shows up in an image? One can of course make a binary decision (appearing or not appearing) based on the similarity between regions in an image and an RP. However, due to inaccurate image segmentation, neither RPs nor regions are free of noise. Thus a binary decision may be very sensitive to such noises. So we propose to use soft decisions based on the idea of fuzzy sets [129]:

- First, for a collection of RPs denoted by $\mathcal{RP} = \{RP_k : k = 1, \cdots, n,\ RP_k = (\mathbf{x}_k^*, \mathbf{w}_k^*)\}$, each RP_k is viewed as a fuzzy set with membership function, $g_k : \mathbb{R}^d \rightarrow [0, 1]$, defined as

$$g_k(\mathbf{x}) = \mu(\|\mathbf{x} - \mathbf{x}_k^*\|_{\mathbf{w}_k^*}) \tag{7.3}$$

  where $\mu(\cdot)$ is a function that is strictly monotonically decreasing on $[0, \infty)$. Therefore, given a region with feature vector $\mathbf{x}$ calculated according to Chapter 5.2, $g_k(\mathbf{x})$, which is called the *degree of membership* of $\mathbf{x}$ with respect to RP_k, illustrates the degree of wellness with which the region belongs to the fuzzy set defined by RP_k. Under the definition (7.3), a region belongs to all RPs with possibly

different degrees of membership. To a certain extent, this models the uncertainties related to image segmentation.

- Next, for an image $\mathbf{B}_i = \{\mathbf{x}_{ij} : j = 1, \cdots, C_i\}$ (where $\mathbf{x}_{ij}$ are region feature vectors), we denote r_k as the degree that RP_k appears in $\mathbf{B}_i$, and define it as

$$r_k = \max_{j=1,\cdots,C_i} g_k(\mathbf{x}_{ij}) \,, \tag{7.4}$$

i.e., the appearance of RP_k in an image is determined by the region that belongs to RP_k with the highest degree of membership. It is clear that r_k is always between 0 and 1. A larger value of r_k indicates a higher degree that RP_k shows up in the image. Binary decision is a special case of the definition (7.4): when $\mu(\cdot)$ is a binary-valued function. Note that, according to (7.3) and (7.4), if $\mu(\cdot)$ is fixed then knowing r_k is equivalent to knowing

$$d_k = \min_{j=1,\cdots,C_i} \|\mathbf{x}_{ij} - \mathbf{x}_k^*\|_{\mathbf{w}_k^*} \,, \tag{7.5}$$

which is the minimum weighted distance from all region feature vectors of an image to RP_k. Since the information of $\mu(\cdot)$ can be implicitly included in the model described below, we use d_k directly instead of r_k to simplify the computation: there is no need to evaluate $\mu(\cdot)$ explicitly.

Now we introduce a rule-based image classifier, which is defined by m rules of the form

$$\text{Rule } j: \quad \text{IF } (d_1 \text{ is } \mathbf{A}_j^1) \text{ AND } (d_2 \text{ is } \mathbf{A}_j^2) \text{ AND } \cdots \text{ AND } (d_n \text{ is } \mathbf{A}_j^n) \text{ THEN } b_j$$

where $\mathbf{A}_j^k$ is a fuzzy set with membership function $a_j^k : \mathbb{R} \to [0,1]$, $j = 1, \cdots, m$, $k = 1, \cdots, n$, b_j is a real number related to class label. Intuitively, $(d_k \text{ is } \mathbf{A}_j^k)$ can be interpreted as "the value of d_k is around some number." Here, the linguistic term "around some number" is mathematically defined by a fuzzy number $\mathbf{A}_j^k$, which can be viewed as a generalized real number. For instance, a fuzzy number $\mathbf{1}$ could be defined by a membership function $\mu_{\mathbf{1}}(x) = e^{-(x-1)^2}$. Given a real number x, $\mu_{\mathbf{1}}(x)$ tells us the degree of membership that x belongs to fuzzy number $\mathbf{1}$ or is "around 1." Under $\mu_{\mathbf{1}}(\cdot)$, a number, which is closer to 1, has a higher degree to be "around 1." Since d_k's are directly related to the degrees that RPs appear in an image, the above rule reasons out a label of an image based on a soft interpretation the appearance of RPs in the image. The question is how to determine $\mathbf{A}_j^k$'s and b_j's. This will be addressed in the next section.

7.3.2 Support Vector Machine Concept Learning

The rule-based classifier introduced in Section 7.3.1 is essentially a fuzzy rule-based system. If we choose product as the fuzzy conjunction operator, addition for fuzzy rule aggregation (it makes the model an additive fuzzy system [60]), and center of area (COA) defuzzification, then the model becomes a special form of the Takagi-Sugeno (TS)

fuzzy model [107]. The input output mapping, $F : \mathbb{R}^n \to \mathbb{R}$, of the model is then defined as

$$F(\mathbf{d}) = \frac{\sum_{j=1}^{m} b_j \prod_{k=1}^{n} a_j^k(d_k)}{\sum_{j=1}^{m} \prod_{k=1}^{n} a_j^k(d_k)}$$

where $\mathbf{d} = [d_1, \cdots, d_n]^T \in \mathbb{R}^n$ is the input. Chapter 4 shows that binary classifiers (multi-class problem can be handled by combining several binary classifiers) can be defined over such a model as

$$label(\mathbf{d}) = \text{sign}(F(\mathbf{d}) + b_0) \tag{7.6}$$

where b_0 is a threshold. Moreover, if we assume that all membership functions associated with the same input variable are generated from location transformation of a reference function, and let $a^k : \mathbb{R} \to [0, 1]$ denote the reference function for $a_j^k(\cdot)$, $j = 1, \cdots, m$ with

$$a_j^k(d_k) = a^k(d_k - z_j^k) \tag{7.7}$$

for some location parameter $z_j^k \in \mathbb{R}$, then the decision function becomes

$$label(\mathbf{d}) = \text{sign}\left(\sum_{j=1}^{m} b_j K(\mathbf{d}, \mathbf{z}_j) + b_0\right) \tag{7.8}$$

where $\mathbf{z}_j = [z_j^1, z_j^2, \cdots, z_j^n]^T \in \mathbb{R}^n$ contains the location parameters of $a_j^k,\ k = 1, \cdots, n$, $K : \mathbb{R}^n \times \mathbb{R}^n \to [0, 1]$ is a kernel defined as

$$K(\mathbf{d}, \mathbf{z}_j) = \prod_{k=1}^{n} a^k(d_k - z_j^k) \quad . \tag{7.9}$$

Table 7.1. A list of positive definite reference functions.

Symmetric Triangle	$\mu(x) = \max(1 - s\,\|x\|, 0), \;\; s > 0$
Gaussian	$\mu(x) = e^{-sx^2}, \;\; s > 0$
Cauchy	$\mu(x) = \frac{1}{1+sx^2}, \;\; s > 0$
Laplace	$\mu(x) = e^{-s\|x\|}, \;\; s > 0$
Hyperbolic Secant	$\mu(x) = \frac{2}{e^{sx}+e^{-sx}}, \;\; s > 0$
Squared sinc	$\mu(x) = \frac{\sin^2(sx)}{s^2x^2}, \;\; s > 0$

This implies that the parameters needed to be learned are m (number of rules), $\mathbf{z}_j$ (location parameters for the IF-part of jth rule), b_j (the THEN-part of jth rule), and b_0 (the threshold).

It is proved in Chapter 4 that the kernel (7.9) becomes a Mercer kernel if the reference functions are positive definite functions [17]. The resulting fuzzy classifier is functionally equivalent to SVMs [115] with kernels defined by (7.9). In particular, each support vector determines the IF-part parameters of one fuzzy rule. The THEN-part parameter is given by the Lagrange multiplier of the support vector. As a result, the proposed rule-based image classifier can be obtained from SVM learning. Many commonly used reference functions are indeed positive definite. An incomplete list is given in Table 7.1. Any convex combination of positive definite functions is still positive definite.

7.3.3 An Algorithmic View

The following pseudo code summarizes the learning process of the proposed rule-based classifier. The input are $\mathcal{D}$ (a collection of bags with binary labels) and $\mathcal{RP}$ (a set

of RPs generated by Algorithm 7.1). The output is an SVM classifier that is functionally equivalent to the proposed rule-based classifier.

Algorithm 7.2. Support Vector Machine Concept Learning

LearnSVM($\mathcal{D}$)

```
1 set S be an empty set
2 FOR (every bag in D)
3     compute d = [d_1, ..., d_n]^T according to (7.5)
4     add (d, Y) to S where Y is the label of the bag
5 END
6 use the given a^k(·)'s to define a kernel function according to (7.9)
7 train an SVM using the data set S and the kernel defined in the
  previous step
8 OUTPUT (the SVM)
```

The above pseudo code assumes that the reference functions $a^k(\cdot)$, $k = 1, \cdots, n$ are given in advance. In our empirical study presented in the next section, different choices of reference functions are compared.

Chapter 8

Experiments

This chapter provides extensive experimental results for the methods proposed in Chapter 5, Chapter 6, and Chapter 7.

8.1 Unified Feature Matching

We implemented the UFM in our experimental SIMPLIcity image retrieval system. The system is tested on a general-purpose image database (from COREL) including about $60,000$ pictures, which are stored in JPEG format with size 384×256 or 256×384. These images were automatically classified into two semantic types: textured photograph, and non-textured photograph [65]. For each image, the features, locations, and areas of all its regions are stored. In Section 8.1.1, we provide several query results on the COREL database to demonstrate qualitatively the accuracy and robustness (to image alterations) of the UFM scheme. Section 8.1.2 presents systematic evaluations of the UFM scheme, and compares the performance of UFM with those of the IRM [64] and EMD-based color histogram [87] approaches based on a subset of the COREL database. The speed of the UFM scheme is compared with that of two other region-based methods in Section 8.1.3. The effect of the choice of membership functions on the performance of the system is presented in Section 8.1.4.

8.1.1 Query Examples

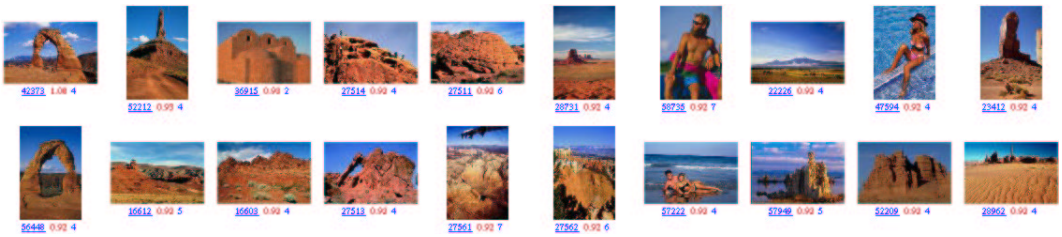
To qualitatively evaluate the accuracy of the system over the 60,000-image COREL database, we randomly pick 5 query images with different semantics, namely natural out-door scene, horses, people, vehicle, and flag. For each query example, we exam the precision of the query results depending on the relevance of the image semantics. We admit that the relevance of image semantics depends on the standpoint of the user. Thus our relevance criteria, specified in Figure 8.1, may be quite different from those used by a user of the system. Due to space limitation, only the top 19 matches to each query are shown in Figure 8.1. We also provide the number of relevant images among top 31 matches. More matches can be viewed from the on-line demonstration site by using the query image ID, given in Figure 8.1, to repeat the retrieval [1].

The robustness of the UFM scheme to image alterations, such as intensity variation, sharpness variation, color distortion, cropping, shifting, rotation, and other intentional distortions, is also tested. Figure 8.2 shows some query results using the 60,000-image COREL database. The query image is the left image for each group of images. In this example, the first retrieved image is exactly the unaltered version of the query image for all tested image alterations except sharpening, in which case, the unaltered version appears in the second place.

8.1.2 Systematic Evaluation

The UFM scheme is quantitatively evaluated focusing on the accuracy, the robustness to image segmentation, and the robustness to image alterations. Comparisons

[1] The demonstration site is at `http://wang.ist.psu.edu/IMAGE`



(a) Natural out-door scene; 15 matches out of 19; 23 matches out of 31

(b) Horses; 19 matches out of 19; 28 matches out of 31

(c) People; 15 matches out of 19; 23 matches out of 31

(d) Vehicle; 17 matches out of 19; 24 matches out of 31

(e) Flag; 19 matches out of 19; 25 matches out of 31

Fig. 8.1. The accuracy of the UFM scheme. For each block of images, the query image is on the upper-left corner. There are three numbers below each image. From left to right they are: the ID of the image in the database, the value of the UFM measure between the query image and the matched image, and the number of regions in the image.

Fig. 8.2. The robustness of the UFM scheme against image alterations.

with the EMD-based color histogram system [87] and the region-based IRM system [64] are also provided. However, it is hard to make objective comparisons with some other region-based searching algorithms such as the Blobworld and the NeTra systems which require additional information provided by the user during the retrieval process.

8.1.2.1 Experiment Setup

To provide more objective comparisons, the UFM scheme is evaluated based on a subset of the COREL database, formed by 10 image categories, each containing 100 pictures. The categories are Africa, Beach, Buildings, Buses, Dinosaurs, Elephants, Flowers, Horses, Mountains, and Food with corresponding Category ID's denoted by integers from 1 to 10, respectively. Within this database, it is known whether any two images are of the same category. In particular, a retrieved image is considered a correct match if and only if it is in the same category as the query. This assumption is reasonable since the 10 categories were chosen so that each depicts a distinct semantic topic. Every image in the sub-database is tested as a query, and the positions of all the retrieval images are recorded.

The following are some notations used in the performance evaluation. $ID(i)$ denotes the *Category ID* of image i ($1 \leq i \leq 1000$ since there are totally 1000 images in the sub-database). It is clear that $ID(i)$ is an integer between 1 and 10 for any $1 \leq i \leq 1000$. For a query image i, $r(i, j)$ is the *rank* of image j (position of image j in the retrieved images for query image i, it is an integer between 1 and 1000). The

precision for query image i, $p(i)$, is defined by

$$p(i) = \frac{1}{100} \sum_{1 \le j \le 1000,\, r(i,j) \le 100,\, ID(j)=ID(i)} 1 \ ,$$

which is the percentage of images belonging to the category of image i in the first 100 retrieved images. Another two statistics are also computed for query image i. They are the *mean rank* $r(i)$ of all the matched images and the *standard deviation* $\sigma(i)$ of the matched images, which are defined by

$$r(i) = \frac{1}{100} \sum_{1 \le j \le 1000,\, ID(j)=ID(i)} r(i,j)$$

and

$$\sigma(i) = \sqrt{\frac{1}{100} \sum_{1 \le j \le 1000,\, ID(j)=ID(i)} [r(i,j) - r(i)]^2} \ .$$

Based on above definitions, we define the *average precision* p_t, *average mean rank* r_t, and *average standard deviation* σ_t for Category t ($1 \le t \le 10$) as

$$p_t = \frac{1}{100} \sum_{1 \le i \le 1000,\, ID(i)=t} p(i), \tag{8.1}$$

$$r_t = \frac{1}{100} \sum_{1 \le i \le 1000,\, ID(i)=t} r(i), \tag{8.2}$$

$$\sigma_t = \frac{1}{100} \sum_{1 \le i \le 1000,\, ID(i)=t} \sigma(i). \tag{8.3}$$

Similarly, the *overall average precision* p, *overall average mean rank* r, and *overall average standard deviation* σ for all images in the sub-database are defined by

$$p = \frac{1}{1000}\sum_{i=1}^{1000} p(i), \tag{8.4}$$

$$r = \frac{1}{1000}\sum_{i=1}^{1000} r(i), \tag{8.5}$$

$$\sigma = \frac{1}{1000}\sum_{i=1}^{1000} \sigma(i). \tag{8.6}$$

Finally, we use entropy to characterize the segmentation-related uncertainties in an image. For image i with C segmented regions, its entropy, $E(i)$, is defined as

$$E(i) = -\sum_{j=1}^{C} P(\mathbf{R}_j^i)\log[P(\mathbf{R}_j^i)], \tag{8.7}$$

where $P(\mathbf{R}_j^i)$ is the percentage of image i covered by region $\mathbf{R}_j^i$. The larger the value of entropy, the higher the uncertainty level. Accordingly, the *overall average entropy* E for all images in the sub-database are define by

$$E = \frac{1}{1000}\sum_{i=1}^{1000} E(i). \tag{8.8}$$

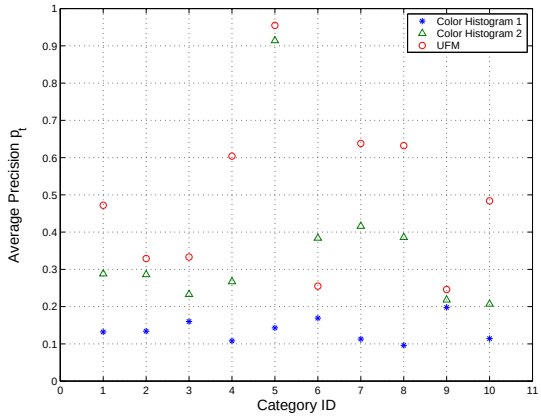
8.1.2.2 Performance on Image Categorization

For image categorization, good performance is achieved when images belonging to the category of the query image are retrieved with low ranks. To that end, the average precision p_t and the average mean rank r_t should be maximized and minimized,

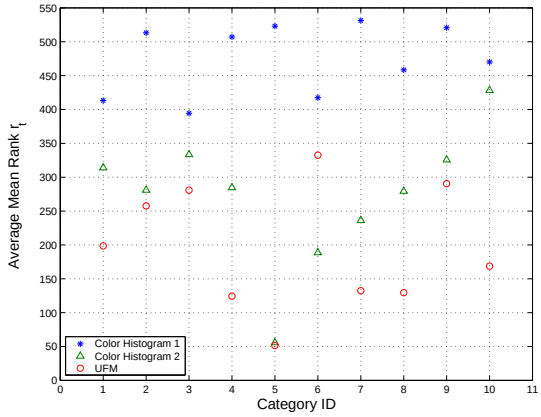
respectively. The best performance, $p_t = 1$ and $r_t = 50.5$, occurs when the first 100 retrieved images belong to Category t for any query image from Category t (since the total number of semantically related images for each query is fixed to be 100). The worst performance, $p_t = 0$ and $r_t = 950.5$, happens when no image in the first 900 retrieved images belongs to Category t for any query image from Category t. For a system that ranks images randomly, p_t is about 0.1, and r_t is about 500 for any Category t. Consequently, the overall average precision p is about 0.1, and the overall average mean rank r is about 500. In the experiments, the recall within the first 100 retrieved images was not computed because it is proportional to the precision in this special case.

The UFM scheme is compared with the EMD-based color histogram matching approach. We use the LUV color space and a matching metric similar to the EMD described in [87] to extract color histogram features and match in the categorized image database. Two different color bin sizes, with an average of 13.1 and 42.6 filled color bins per image, are evaluated. we call the one with less filled color bins the Color Histogram 1 system and the other the Color Histogram 2 system. Comparisons of average precision p_t, average mean rank r_t, and average standard deviation σ_t are given in Figure 8.3. p_t, r_t, and σ_t are computed according to equations (8.1), (8.2), and (8.3), respectively.

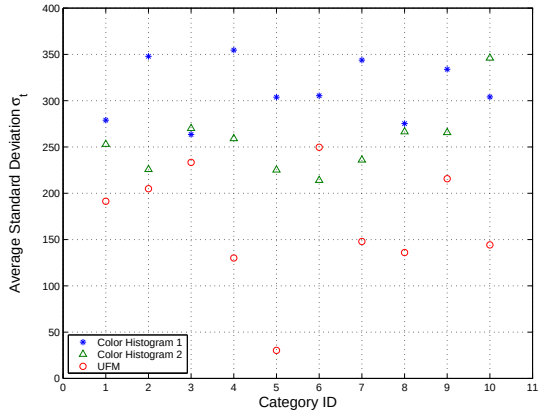
It is clear that the UFM scheme performs much better than both of the two color histogram-based approaches in almost all image categories. The performance of the Color Histogram 2 system is better that that of the Color Histogram 1 system due to more detailed color separation obtained with more filled bins. However, the price paid for the performance improvement is the decrease in speed. The UFM runs at about



Average precision p_t



Average mean rank r_t



Average standard deviation σ_t

Fig. 8.3. Comparing the UFM scheme with the EMD-based color histogram approaches on average precision p_t, average mean rank r_t, and average standard deviation σ_t. For p_t, the larger numbers indicate better results. For r_t and σ_t, the lower numbers denote better results.

twice the speed of the relatively fast Color Histogram 1 system and still provides much better retrieval accuracy than the extremely slow Color Histogram 2 system.

The UFM scheme is also compared with the IRM approach [64] using the same image segmentation algorithm with the average number of regions per image for all images in the sub-database being 8.64. Experiment results show that the UFM scheme outperforms the IRM approach by a 6.2% increase in overall average precision, a 6.7% decrease in the overall average mean rank, and a 4.0% decrease in the overall average standard deviation.

8.1.2.3 Robustness to Segmentation-Related Uncertainties

Because image segmentation cannot be perfect, being robust to segmentation-related uncertainties becomes a critical performance index for a region-based image retrieval system. In this section, we compare the performance of the UFM and IRM approaches with respect to the coarseness of image segmentation. We use the entropy, defined by equation (8.7), to measure the segmentation-related uncertainty levels. As we will see, the overall average entropy E, given by (8.8), increases with the increase of the *average number of regions* C for all images in the sub-database. Thus, we can adjust the average uncertainty level through changing the value of C. The control of C is achieved by modifying the stop criteria of the k-means algorithm. Figure 8.4 shows two images, beach scene and bird, and the segmentation results with different number of regions. Segmented regions are shown in their representative colors. Segmentation results for all images in the database can be found on the demonstration web site.

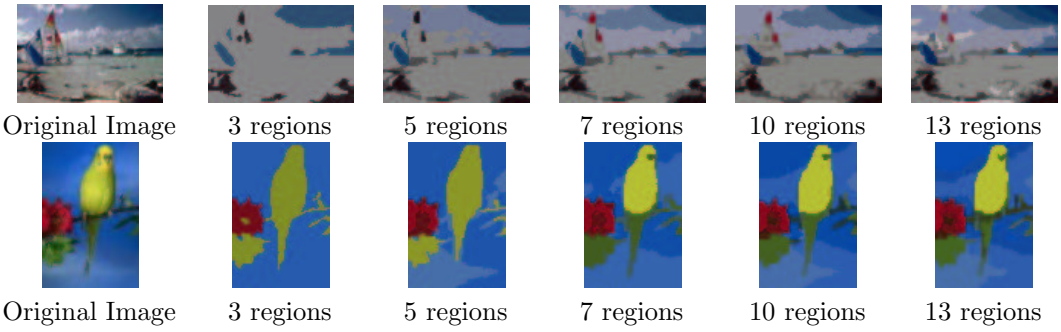


Fig. 8.4. Segmentation results by the k-means clustering algorithm. Original images are in the first column.

To give a fair comparison between UFM and IRM at different uncertainty levels, we perform the same experiments for different values of C (4.31, 6.32, 8.64, 11.62, and 12.25). Based on equations (8.4), (8.5), and (8.6), the performance in terms of overall average precision p, overall average mean rank r, and overall average standard deviation σ are evaluated for both approaches. The results are given in Figure 8.5. As we can see, the overall average entropy E increases when images are, on average, segmented into more regions. In other words, the uncertainty level increases when segmentation becomes finer. At all uncertainty levels, the UFM scheme performs better than the IRM method in all three statistics, namely p, r, and σ. In addition, there is a significant increase in p and a decrease in r for the UFM scheme as the average number of regions increases. While for the IRM method, p and r almost remain unchanged for all values of C. This can be explained as follows. When segmentation becomes finer, although the uncertainty level increases, more details (or information) about the original image are also preserved (as shown in Figure 8.4). Compared with the IRM method, the UFM

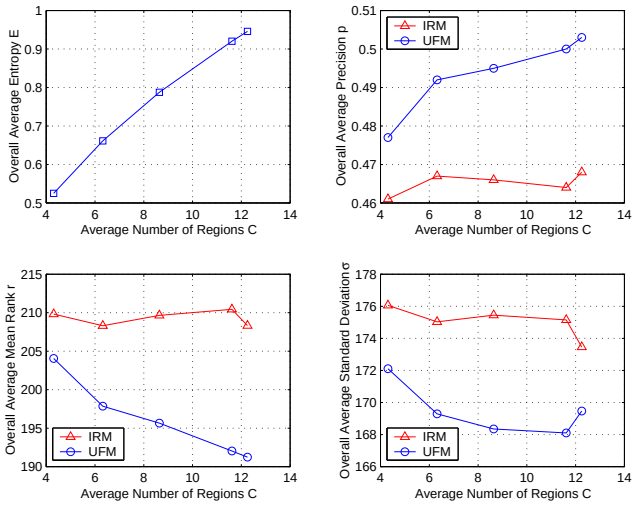


Fig. 8.5. Comparing the UFM scheme with the IRM method on the robustness to image segmentation: overall average entropy E, overall average precision p, overall average mean rank r, and overall average standard deviation σ.

scheme is more robust to segmentation-related uncertainties and thus benefits more from the increasing of the average amount of information per image.

8.1.2.4 Robustness to Image Alterations

The UFM approach has been tested for the robustness to image alterations including intensity variation, color distortion, sharpness variation, shape distortion, cropping, and shifting. The goal is to demonstrate the ability of the system to recognize an image when its altered version is submitted as the query. We apply image alteration to an image (called target image i) in the sub-database. The resulting image i' is then used as the query image, and the rank of the retrieved target image i, $r(i', i)$, is recorded. Repeating the process for all images in the sub-database, the average rank r' for target

images and the standard deviation σ' of the rank are computed as

$$r' = \frac{1}{1000}\sum_{i=1}^{1000} r(i', i) \tag{8.9}$$

$$\sigma' = \sqrt{\frac{1}{1000}\sum_{i=1}^{1000}\left[r(i', i) - r'\right]^2}. \tag{8.10}$$

Clearly, smaller numbers for r' and σ' indicate more robust performance.

For each type of image alteration, curves for r' and σ' with respect to the intensity of image alteration are plotted in Figure 8.6. If we call a system being robust to image alterations when the target image appear in the first 10 retrieved images, then, on average, the UFM scheme is robust to approximately 22% brightening, 20% darkening, 56% more saturation, 30% less saturation, 5×5 Gaussian filter, random spread pixels in a 14×14 neighborhood, and cropping 45%. The UFM scheme is extremely robust to horizontal and vertical image shifting.

8.1.3 Speed

The algorithm has been implemented on a Pentium III 700MHz PC running Linux operating system. Computing the feature vectors for $60,000$ color images of size 384×256 requires around 17 hours. On average, one second is needed to segment and compute the fuzzy features for an image, which is the same as the speed of IRM. It is much faster than the Blobworld system [10], which, on average, takes about 5 minutes to segment a

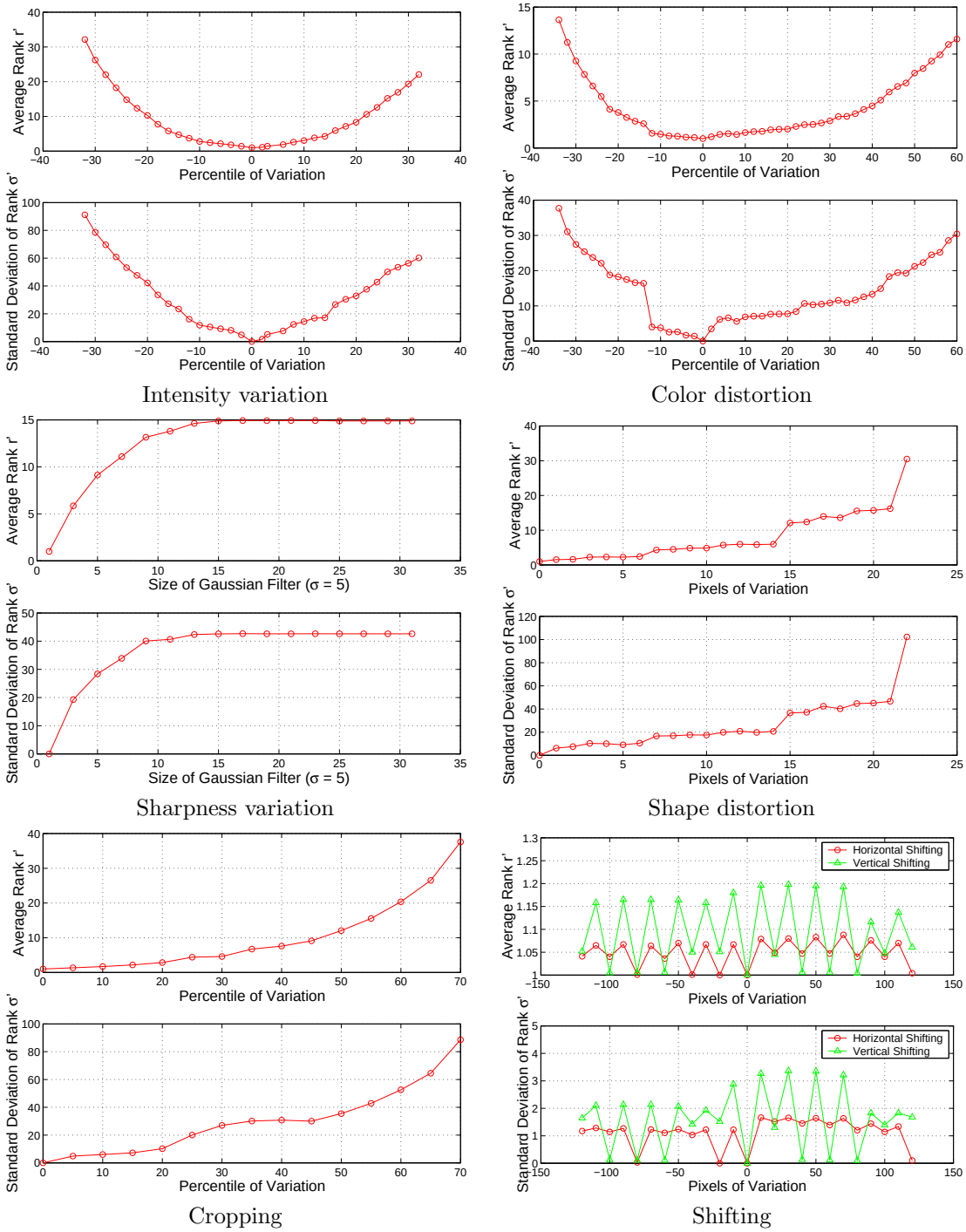


Fig. 8.6. The robustness of the UFM scheme to image alterations. Average rank r' and standard deviation of rank σ' are plotted against the intensity of image alterations.

128×192 image [2]. Fast segmentation speed provides us the ability of handling outside queries in real-time.

The time for matching images and sorting results in UFM is $O(C^2N + N \log N)$, where N is the number of images in the database, C is the average number of regions of an image. For our current database ($N = 60,000$ and $C = 4.3$), when the query image is in the database, it takes about 0.7 seconds of CPU time on average to compute and sort the similarities for all images in the database. If the query is not in the database, one extra second of CPU time is spent to process the query.

Based on 100 random runs, a quantitative comparison of the speed of UFM, IRM, and Blobworld systems is summarized in Table 8.1 where t_s is the average CPU time for image segmentation, t_i is the average CPU time for computing similarity measures and indexing [3]. The UFM and IRM use the same database of $60,000$ images. The Blobworld system is tested on a database of $35,000$ images. Unlike IRM and UFM, the Blobworld system doesn't support outside queries. For inside queries, which do not require online image segmentation, UFM is 0.43 times faster than IRM, and 6.57 times faster than Blobworld.

8.1.4 Comparison of Membership Functions

The UFM scheme is tested against different membership functions, namely the cone, exponential, and Cauchy functions. To make comparisons consistent, for a given

[2]The segmentation algorithm (in Matlab code) is tested on a 400MHz UltraSPARC IIi with the code obtained from `http://elib.cs.berkeley.edu/src/blobworld/`.

[3]Approximate execution times are obtained by issuing queries to the demonstration web sites `http://wang.ist.psu.edu/IMAGE/` (UFM and IRM) and `http://elib.cs.berkeley.edu/photos/blobworld/` (Blobworld). The web server for UFM and IRM is a 700MHz Pentium III PC, while the web server for Blobworld is unknown.

Table 8.1. Comparison of UFM, IRM, and Blobworld systems on average segmentation time t_s and average indexing time t_i.

CBIR Systems	**UFM**	**IRM**	**Blobworld**
t_s(second)	1.1	1.1	292.5
t_i(second)	0.7	1.0	5.3
Database Size	60,000	60,000	35,000

region, we require the fuzzy features with different membership functions have identical 0.5-cuts. The 0.5-cut of a fuzzy feature is the set of feature vectors that have degrees of membership greater than or equal to 0.5. For a Cauchy function $\mathcal{C}(\mathbf{x}) = \frac{d^\alpha}{d^\alpha + \|\mathbf{x}-\mathbf{v}\|^\alpha}$, the above requirement can be easily satisfied by choosing the cone function as $\mathcal{T}(\mathbf{x}) = \max(1 - \frac{\|\mathbf{x}-\mathbf{v}\|^\alpha}{(2d)^\alpha}, 0)$ and the exponential function as $\mathcal{E}(\mathbf{x}) = e^{-\frac{\|\mathbf{x}-\mathbf{v}\|^\alpha}{(1.443d)^\alpha}}$.

Under an experiment setup identical to that of Section 8.1.2.2, the performance on image categorization is tested for three membership functions with parameter α varying from 0.1 to 2.0. The overall average precision p is calculated according to (8.4). As shown in the upper plot in Figure 8.7, the highest p for Cauchy and exponential membership functions, which is 0.477, occurs at $\alpha = 1.0$. The best α for the cone membership function is 0.8 with $p = 0.478$. So three membership functions generate almost the same maximum overall average precision. However, the computational complexities of three membership functions with corresponding optimal α values are quite different. For any given $\|\mathbf{x}-\mathbf{v}\|$, the cone membership function needs to compute a power term $\left(\frac{\|\mathbf{x}-\mathbf{v}\|}{2d}\right)^{0.8}$. The exponential membership function needs to evaluate an exponential term $e^{-\frac{\|\mathbf{x}-\mathbf{v}\|}{1.443d}}$. Only two floating point operations are required by the Cauchy membership function.

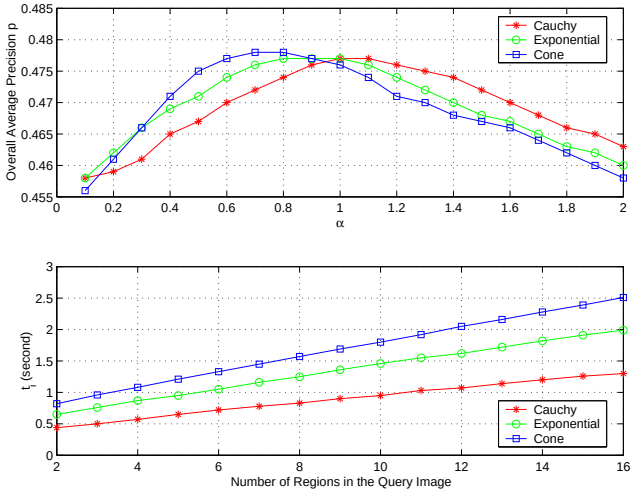


Fig. 8.7. Comparing the Cauchy, exponential, and cone membership functions on overall average precision p and average CPU time t_i for inside queries.

Based on the 60,000-image database, t_i for three membership functions are plotted in the lower part of Figure 8.7. As expected, t_i enlarges linearly with the increase of the number of regions in the query image and the Cauchy membership function produces the smallest t_i.

We also test the robustness to image alterations with respect to the type of membership function being used. For all six image alterations described in Section 8.1.2.4, the performances of exponential ($\alpha = 1.0$) and cone ($\alpha = 0.8$) membership functions are almost identical to that of the Cauchy ($\alpha = 1.0$) membership function in terms of r' and σ' defined by (8.9) and (8.10), respectively. The Cauchy membership function requires the least computational cost.

8.2 Cluster-Based Retrieval of Images

Our system is implemented with the same general-purpose image database as in Section 8.1. In Section 8.2.1, we provide several query results on the COREL database to intuitively illustrate the performance of the system. Section 8.2.2 presents systematic evaluations of CLUE algorithm in terms of the goodness of image clustering and retrieval accuracy. Numerical comparisons with the SIMPLIcity system using UFM similarity measure are also given. In Section 8.2.3, the speed of CLUE is compared with that of a typical CBIR system using UFM similarity measure. The influence of k and r parameters in NNM on the performance of the system is presented in Section 8.2.4. Section 8.2.5 presents some preliminary results on images returned by Google's Image Search.

8.2.1 Query Examples

To qualitatively evaluate the performance of the system over the $60,000$-image COREL database, we randomly pick five query images with different semantics, namely, *birds*, *car*, *food*, *historical buildings*, and *soccer game*. For each query example, we examine the precision of the query results depending on the relevance of the image semantics. Here only images in the first cluster, in which the query image resides, are considered. This is because images in the first cluster can be viewed as sharing the same similarity-induced semantics as that of the query image according to the clusters organization described in Section 6.3.2. Performance issues about the rest clusters will be covered in Section 8.2.2. Since CLUE of our system is built upon UFM similarity measure, query results of a typical CBIR system, SIMPLIcity system using UFM similarity measure (we

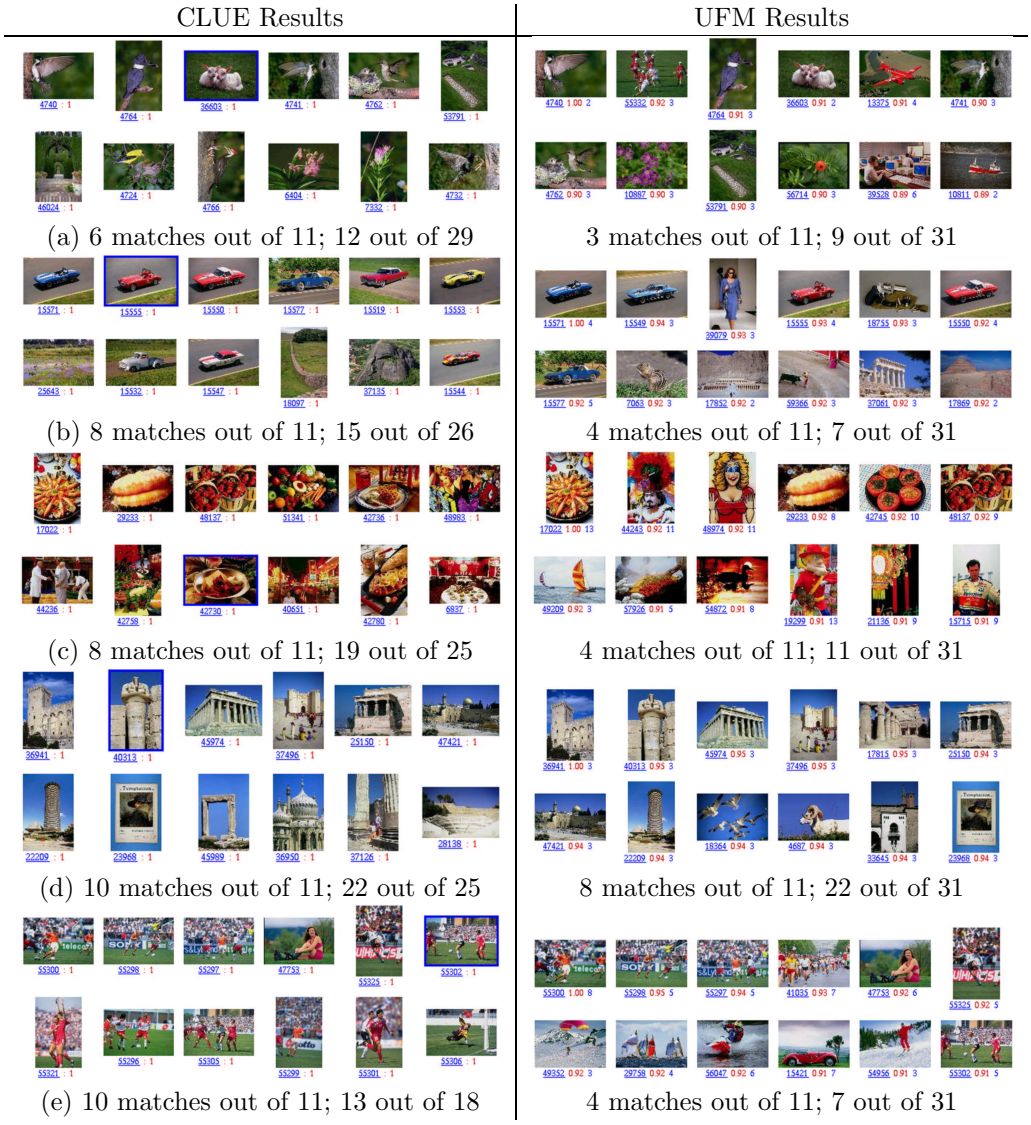


Fig. 8.8. Comparison of CLUE and UFM. The query image is the upper-left corner image of each block of images. The underlined numbers below the images are the ID numbers of the images in the database. For the images in the left column, the other number is the cluster ID (the image with a border around it is the representative image for the cluster). For images in the right column, the other two numbers are the value of UFM measure between the query image and the matched image, and the number of regions in the image. (a) birds, (b) car, (c) food, (d) historical buildings, and (e) soccer game.

call the system UFM to simplify notation), are also included for comparison. We admit that the relevance of image semantics depends on standpoint of a user. Therefore, our relevance criteria, specified in Figure 8.8, may be quite different from those used by a user of the system. Due to space limitations, only the top 11 matches to each query are shown in Figure 8.8. We also provide the number of relevant images in the first cluster (for CLUE) or among top 31 matches (for UFM).

Compared with UFM, CLUE provides semantically more precise results for all query examples given in Figure 8.8. This is reasonable since CLUE utilizes more information about image similarities than UFM does. CLUE groups images into clusters based on pairwise distances so that the within-cluster similarity is high and between-cluster similarity is low. The results seem to indicate that a similarity-induced image cluster tends to contain images of similar semantics. In other words, organizing images into clusters and retrieving image clusters may help to reduce the semantic gap even when the rest of the components of the system, such as feature extraction and image similarity measure, remain unchanged.

8.2.2 Systematic Evaluation

To provide a more objective evaluation and comparison, CLUE (built upon UFM similarity measure) is tested on a subset of the COREL database, formed by 10 image categories, each containing 100 images. The categories are Africa people and villages, Beach, Buildings, Buses, Dinosaurs, Elephants, Flowers, Horses, Mountains and glaciers, and Food with corresponding Category IDs denoted by integers from 1 to 10, respectively. Within this database, it is known whether two images are of the same category (or

semantics). Therefore we can quantitatively evaluate and compare the performance of CLUE in terms of the goodness of image clustering and retrieval accuracy. In particular, the goodness of image clustering is measured via the distribution of images semantics in the cluster, and a retrieved image is considered a correct match if and only if it is the same category as the query image. These assumptions are reasonable since the 10 categories were chosen so that each depicts a distinct semantic topic.

8.2.2.1 Goodness of Image Clustering

Ideally, a CBICR system would be able to generate image clusters each of which contains images of similar or even identical semantics. The *confusion matrix* is one way to measure clustering performance. However, to compute the confusion matrix, the number of clusters needs to be equal to the number of distinct semantics, which is unknown in practice. Although we can force CLUE to always generate 10 clusters in this particular experiment, the experiment setup would then be quite different to a real application. So we use *purity* and *entropy* to measure the goodness of image clustering.

Assume we are given a set of n images belonging to c distinctive categories (or semantics) denoted by $1, \cdots, c$ (in this experiment $c \leq 10$ depending on the collection of images generated by NNM) while the images are grouped into m clusters $\mathbf{C}_j$, $j = 1, \cdots, m$. Cluster $\mathbf{C}_j$'s purity can be defined as

$$p(\mathbf{C}_j) = \frac{1}{|\mathbf{C}_j|} \max_{k=1,\cdots,c} |\mathbf{C}_{j,k}| \tag{8.11}$$

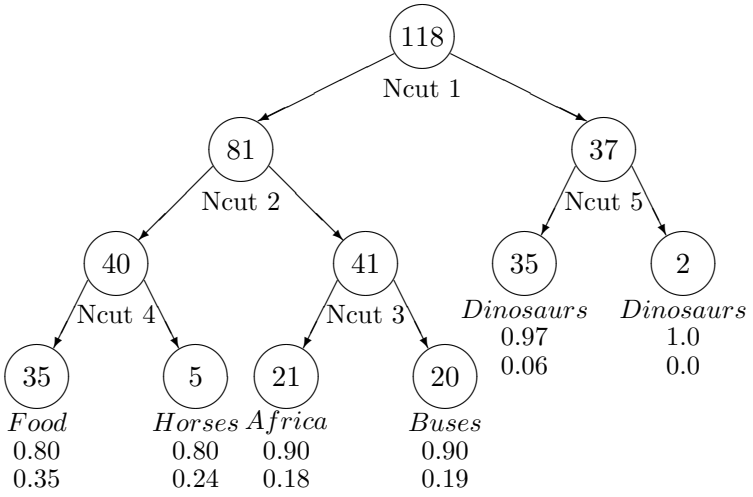


Fig. 8.9. CLUE applies five Ncuts to a collection of 118 images neighboring to a query image of *food*. Numbers within each node denote the size of the corresponding clusters. Linguistic descriptor and numbers listed under each leaf node are (from top to bottom): name of the dominant semantic category in the leaf node (or cluster), purity of the cluster, and entropy of the cluster.

where $\mathbf{C}_{j,k}$ consists of images in $\mathbf{C}_j$ that belong to category k, and $|\mathbf{C}_j|$ represents the size of the set. Each cluster may contain images of different semantics. Purity gives the ratio of the dominant semantic class size in the cluster to the cluster size itself. The value of purity is always in the interval $[\frac{1}{c}, 1]$ with a larger value means that the cluster is a "purer" subset of the dominant semantic class. Entropy is another cluster quality measure, which is defined as follows:

$$h(\mathbf{C}_j) = -\frac{1}{\log c} \sum_{k=1}^{c} \frac{|\mathbf{C}_{j,k}|}{|\mathbf{C}_j|} \log \frac{|\mathbf{C}_{j,k}|}{|\mathbf{C}_j|} \ . \tag{8.12}$$

Since entropy considers the distribution of semantic classes in a cluster, it is a more comprehensive measure than purity. Note that we have normalized entropy so that the value is between 0 and 1. Contrary to the purity measure, an entropy value near 0 means the cluster is comprised mainly of 1 category, while an entropy value close to 1 implies that the cluster contains a uniform mixture of all categories. For example, if half of the images of a cluster belong to one semantic class and the rest of the images are evenly divided into 9 different semantic classes, then the entropy is 0.7782 and the purity is 0.5. Figure 8.9 shows clusters and the associated tree structure generated by CLUE for a sample query image of food. Size of each cluster, purity and entropy of leaf clusters are also listed.

The following are some additional notations used in the performance evaluation. For a query image i: 1) m_i denotes the *number of retrieved clusters*; 2) v_i is the *average size* of the retrieved clusters; 3) $P(i)$ is the *average purity* of the retrieved clusters, i.e., $P(i) = \frac{1}{m_i} \sum_{j=1}^{m_i} p(\mathbf{C}_j)$ where $p(\mathbf{C}_j)$ is computed according to (8.11); and 4) $H(i)$ is the

average entropy of the retrieved clusters, i.e., $H(i) = \frac{1}{m_i}\sum_{j=1}^{m_i} h(\mathbf{C}_j)$ where $h(\mathbf{C}_j)$ is computed according to (8.12).

Every image in the 1000-image database is tested as a query. The same set of parameters specified in Section 6.3.4 is used here. For query images within one semantic category, the following statistics are computed: the mean of m_i, the mean and standard deviation (STDV) of v_i, the mean of $P(i)$, and the mean of $H(i)$. In addition, we calculate P_{NNM} and H_{NNM} for each query, which are respectively the purity and entropy of the whole collection of images generated by NNM, and the mean of P_{NNM} and H_{NNM} for query images within one semantic category. The results are summarized in Table 8.2 (second and third columns) and Figure 8.10. The third column of Table 8.2 shows that the size of clusters does not vary greatly within a category. This is because of the heuristic used in recursive Ncut: always dividing the largest cluster. It should be observed from Figure 8.10 that CLUE provides good quality clusters in the neighborhood of a query image. Compared with the purity and entropy of collections of images generated by NNM, the quality of the clusters generated by recursive Ncut is on average much improved for all image categories except category 5, for which NNM generates quite pure collections of images leaving little room for improvement.

8.2.2.2 Retrieval Accuracy

For image retrieval, purity and entropy by themselves may not provide a comprehensive estimate of the system performance even though they measure the quality of image clusters. Because what could happen is a collection of semantically pure image clusters but none of them sharing the same semantics with the query image. Therefore

Table 8.2. Statistics of the average number of clusters m_i and the average cluster size v_i, and an estimation of the correct categorization rate C_t.

ID. Category Name	Mean m_i	Mean $v_i \pm$ STDV	C_t
1. Africa people and villages	7.77	14.0 ± 3.80	0.75
2. Beach	7.96	13.6 ± 2.11	0.55
3. Buildings	7.89	11.8 ± 3.81	0.69
4. Buses	7.88	8.61 ± 3.49	0.88
5. Dinosaurs	7.96	6.51 ± 0.68	1.00
6. Elephants	7.52	14.6 ± 3.94	0.64
7. Flowers	8.00	8.84 ± 1.79	0.95
8. Horses	8.00	9.98 ± 2.95	0.97
9. Mountains and glaciers	7.84	14.0 ± 2.70	0.51
10. Food	7.79	12.2 ± 2.48	0.78

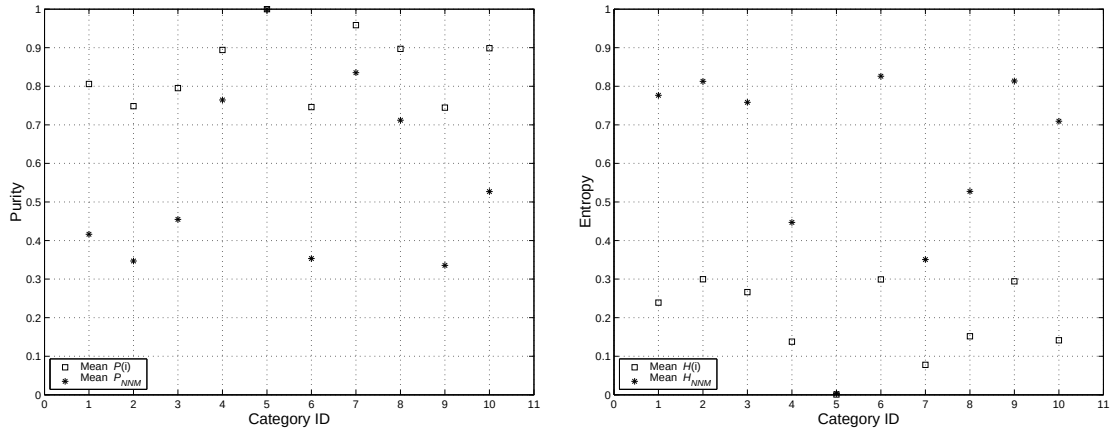


Fig. 8.10. Clustering performance in terms of purity and entropy. For mean $P(i)$ and mean P_{NNM}, larger numbers indicate purer clusters. For mean $H(i)$ and mean H_{NNM}, smaller numbers denote better cluster quality.

one needs to consider the semantic relationship between these image clusters and the query image. For this purpose, we introduce the *correct categorization rate* and *average precision*.

A query image is correctly categorized if the dominant category in the query image cluster (first cluster of leftmost leaf) is identical to the query category. The correct categorization rate, C_t, for image category t indicates how likely the dominant semantics of the query image cluster coincides with the query semantics, and is defined as the ratio of the number of correctly categorized images in category t to the size of category t. The fourth column of Table 8.2 lists values of C_t for 10 categories used in our experiments. Note that randomly assigning a dominant category to the query image cluster will give a C_t value of 0.1. The results there indicate that CLUE has some difficulties in categorizing images about beaches (category 2) and images about mountains and glaciers (category 9), even though the performance is still four times better than random. A detailed examination of the errors shows that most errors on these two categories are errors between these two categories, i.e., a beach query is categorized as mountains and glaciers, or conversely. The performance degradation on these two categories seems understandable. Many images from these two categories are visually similar. Some beach images contain mountains or mountain-like regions, while some mountain images have regions corresponding to river, lake, or even ocean. In addition, UFM measure may also mistakenly view a glacier as clouds because both regions have similar white color and shape. However, we argue that the performance may be improved if a better similarity measure is used.

From the standpoint of a system user, the correct categorization rate may not be the most important performance index. Even if the first cluster, in which the query image resides, does not contain any images that are semantically similar to the query image, the user can still look into the rest of the clusters. So we use *precision* to measure how likely a user would find images belonging to the query category within a certain number of top matches. Here the precision is computed as the percentage of images belonging to the category of the query image in the first 100 retrieved images. The *recall* equals precision for this special case since each category has 100 images. The r parameter in NNM is set to be 30 to ensure that the number of neighboring images generated is greater than 100. As mentioned in Section 6.3.2, the linear organization of clusters may be viewed as a structured sorting of clusters in ascending order of distances to a query image (recall that images within each cluster are organized in ascending order of distances to the query). Therefore the top 100 retrieved images are found according to the order of clusters. The *average precision* for a category t is then defined as the mean of precision for query images in category t. Figure 8.11 compares the average precision given by CLUE with those obtained by UFM. Clearly, CLUE performs better than UFM for 9 out of 10 categories (they tie on the remaining one category). The overall average precision for 10 categories are 0.538 for CLUE and 0.477 for UFM. We want to emphasize again: CLUE can be built upon any real-valued symmetric similarity measure, not just UFM similarity measure. The results here suggest that on average CLUE scheme may improve the precision of a CBIR system.

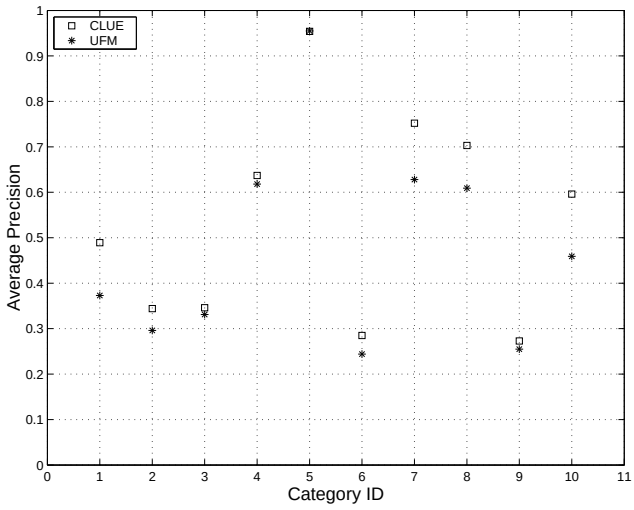


Fig. 8.11. Comparing CLUE scheme with UFM method on the average precision.

8.2.3 Speed

The CLUE has been implemented on a Pentium III 700MHz PC running Linux operation system. To compare the speed of the CLUE with the UFM, which is implemented and tested on the same computer, 100 random queries are issued to the demonstration web sites. The CLUE takes on average 0.8 second per query for similarity measure evaluation, sorting, and clustering, while the UFM takes 0.7 second to evaluate similarities and sort the results. The size of the database is $60,000$ for both tests. Although the CLUE is slower than the UFM because of the extra computational cost for NNM and recursive Ncut, the execution time is still well within the tolerance of real-time image retrieval.

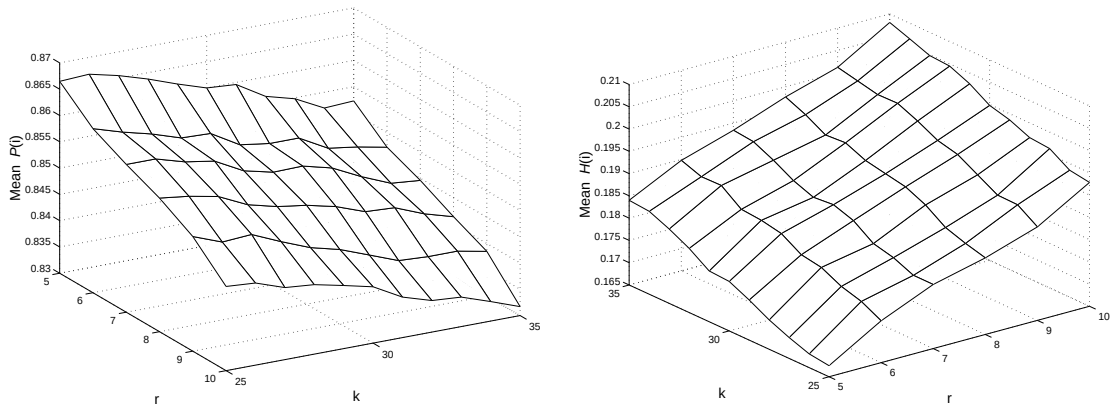


Fig. 8.12. Robustness to the number of neighboring images: mean $P(i)$ and mean $H(i)$ over 1000 query images for different values of k and r.

8.2.4 Robustness

CLUE is tested for the robustness to the number of neighboring images, which is decided by k and r parameters for NNM. Given a fixed pair of k and r, the average purity $P(i)$ and average entropy $H(i)$ for each image in the 1000-image database are calculated. Then we compute the mean of $P(i)$ and the mean of $H(i)$ over all 1000 query images. The same steps are repeated for different pairs of k and r where k and r take values from $\{25, 26, \cdots, 35\}$ and $\{5, 6, \cdots, 10\}$, respectively. The resulting mean purity and mean entropy are shown in Figure 8.12. For 66 different pairs of k and r, the mean $P(i)$ varies within the interval $[0.832, 0.867]$ and the mean $H(i)$ varies within the interval $[0.168, 0.208]$. Considering that the average number of neighboring images varies from 59 to 116 (the average numbers of neighboring images are around 59 and 116 for $(k, r) = (25, 5)$ and $(35, 10)$, respectively), the variations on the purity and entropy are not significant.

8.2.5 Results on WWW Images

To show the performance of CLUE on real world image data, we provide some preliminary results using images crawled from the Internet. The images are obtained from Google's Image Search (`http://images.google.com`), which is a keyword-based image retrieval system. Due to space limitation, we only present the results for two query words: Tiger and Beijing. Since there is no query image, the neighboring image selection stage of CLUE is skipped. Instead, for each query word, the recursive Ncut using the same set of parameters as in the above experiments is directly applied to the top 200 images returned by Google. Figure 8.13 lists some sample images from the top 4 largest clusters for each query word. Each block of images are chosen to be the top 18 images within a cluster that are closest to the representative image of the cluster in terms of UFM similarity measure. The cluster size is also specified below each block of images.

As shown in Figure 8.13, real world images can be visually and semantically quite heterogeneous even when a very specific category is under consideration. The Tiger images returned by Google's Image Search contains images of cartoon tiger (animal), real tiger (animal), Tiger Woods (golf player), Tiger tank, Crouching Tiger Hidden Dragon (movie), and tiger shark, etc. Images about Beijing include images of city maps, people, and buildings, etc. CLUE seems to be capable of providing visually coherent image clusters with reduced semantic diversity within each cluster. The images in Figure 8.13(a) are mainly about cartoon tigers. Half of the images in Figure 8.13(d) contain people. Real tigers appear more frequently in Figure 8.13(b) and (c) than in

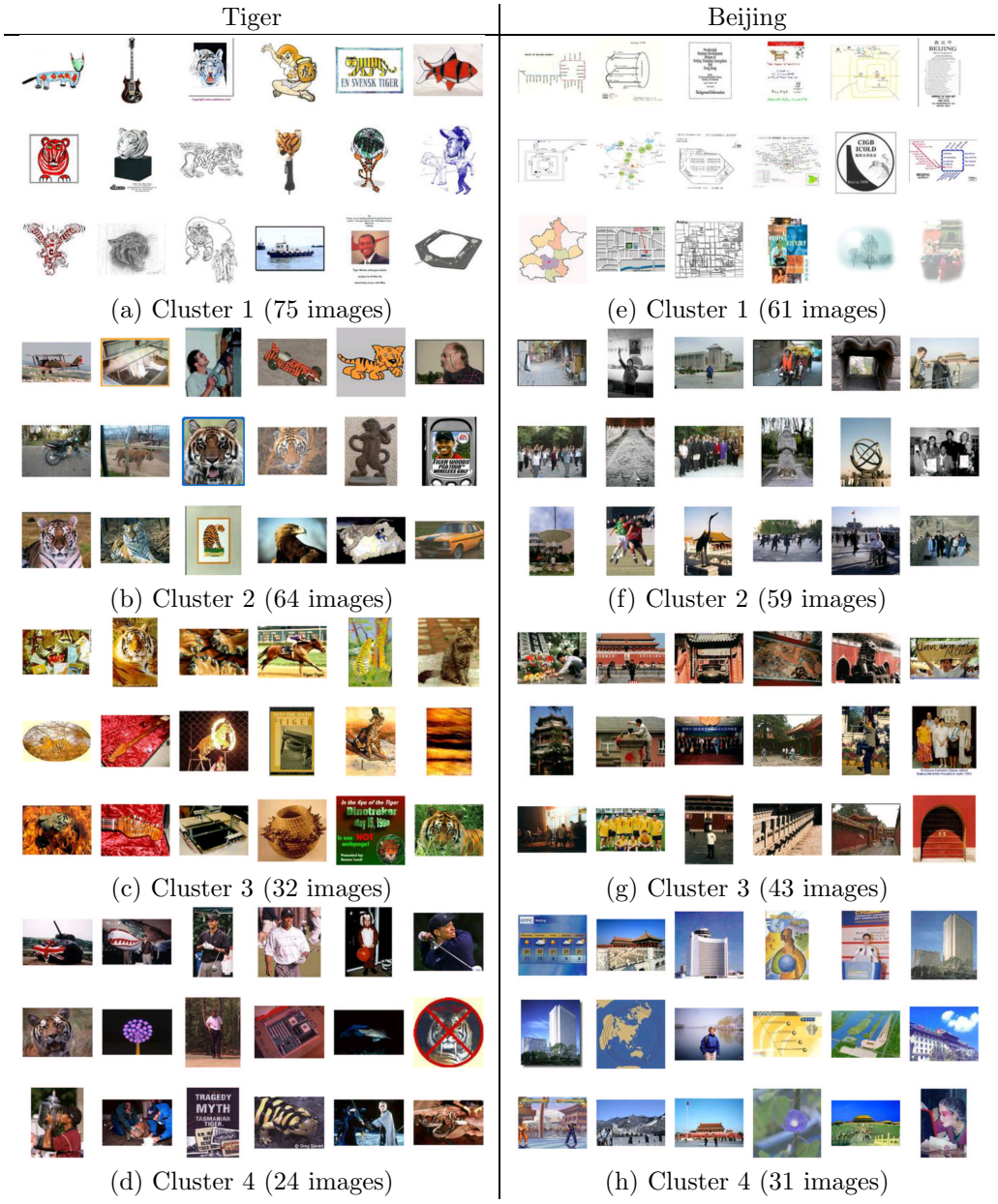


Fig. 8.13. Some sample images of the top four largest clusters obtained by applying CLUE to images returned by Google's Image Search with query words Tiger (left column) and Beijing (right column).

Figure 8.13(a) and (b). Images in Figure 8.13(c) have stronger textured visual effect than images of the other three blocks. The remaining 5 images (four largest clusters of Tiger take 195 images of the total 200 images), which are not included in the figure, are all about tiger sharks. As to images about Beijing, the majority of the images in Figure 8.13(e) are city maps. Out of the 18 images in Figure 8.13(f), 11 contains people. The majority of images in Figure 8.13(g) are about Beijing's historical buildings. There also a lot of images of buildings in Figure 8.13(h). But most of them are modernbuilt. These results seem to imply that, to some extent, unsupervised learning is helpful in disambiguating and refining image semantics and may improve the performance of a keyword-based image retrieval system.

8.3 Image Categorization

In this section we present systematic evaluations of the image categorization method proposed in Chapter 7 based on a collection of images from COREL. Section 8.3.1 describes the experiment setup including image dataset, implementation details, and parameters selection. Section 8.3.2 compares the classification accuracies of the proposed approach (using different reference functions) with those of two image classification methods. The effect of inaccurate image segmentation on classification accuracies is demonstrated in Section 8.3.3. Section 8.3.4 illustrates the performance variations when the number of categories in a dataset increases. The computational issues are discussed in Section 8.3.5.

8.3.1 Experiment Setup

The dataset used in our empirical study consists of 2000 images from the COREL database used in Section 8.1 and 8.2. They belong to 20 thematically diverse image categories, each containing 100 images. The category names and some randomly selected sample images from all 20 categories are shown in Figure 8.14. As we can see, images within each category are not necessarily all visually similar. While images from different categories may be visually similar to each other.

Images within each category are randomly splitted into a training set and a test set each with 50 images. We repeat each experiment for 5 random splits, and report the average (and the standard deviation) of the results obtained over 5 different test sets. The SVM^{Light} [55] software is used to train the SVMs. The classification problem here is clearly a multi-class problem. We use the one-against-the-rest approach: 1) For each category, an SVM is trained to separate that category from all the rest categories; 2) The final predicted class label is decided by the winner of all SVMs, i.e., one with the maximum un-thresholded output.

Two other image classification methods are implemented for comparison. One is a histogram-based SVM classification approach proposed in [13] (we denote it as Hist-SVM). Each image is represented by a color histogram in the LUV color space. The dimension of each histogram is 125. The other is an SVM-based MIL method introduced in [2] (we call it MI-SVM). Since MI-SVM is identical to our approach in terms of image representation (both are built on features of segmented regions), same

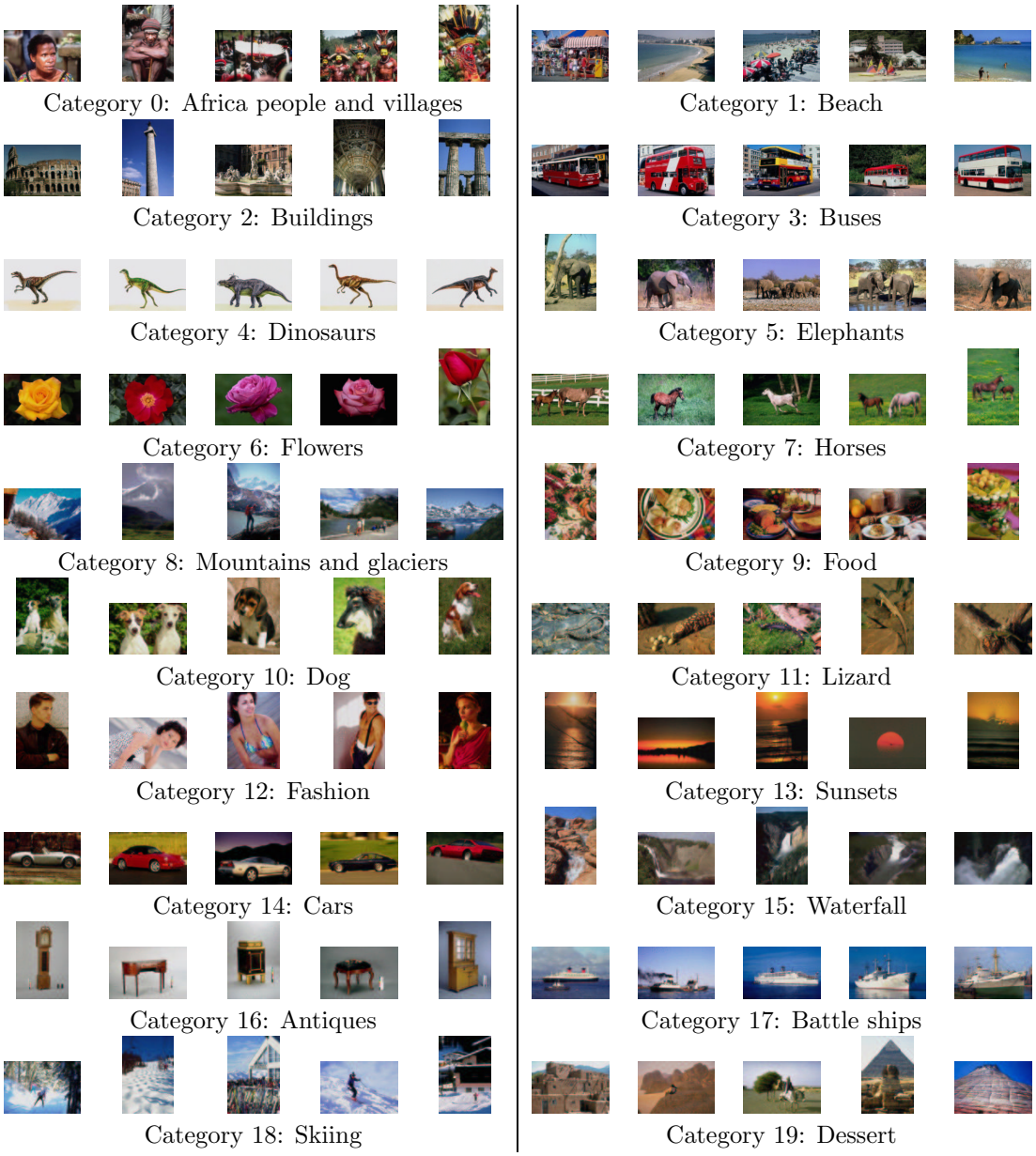


Fig. 8.14. Sample images taken from 20 categories.

image representation described in Section 5.2 are used by both methods. The learning problems in Hist-SVM and MI-SVM are solved by SVM^{Light}.

Several parameters need to be specified for SVM^{Light} [4]. The most significant ones are the trade-off between training error and margin, the type of kernel functions, and kernel parameter. We apply the following strategy to select these parameters:

- First, we pick the type of kernel functions. For our proposed method, the kernel function is determined by reference functions. Different choices of reference functions will be tested and compared. For Hist-SVM and MI-SVM, we choose Gaussian kernel.

- Then we allow each one of the trade-off parameter and kernel parameter (for our proposed method, the kernel parameter is the constant s in Table 7.1) be respectively chosen from two sets each containing 10 predetermined numbers. For every pair of values of the two parameters (there are 100 pairs in total), a twofold cross-validation error on the training set is recorded. The pair that gives the minimum twofold cross-validation error is selected to be the "optimal" parameters.

Note that the above procedure is applied only once for each method. Once the parameters are determined, the learning is performed over the whole training set.

8.3.2 Categorization Results

The classification results provided in this section are based on images in Category 0 to Category 9, i.e., 1000 images. Results for the whole dataset will be given in

[4] SVM^{Light} software and detailed descriptions of all its parameters are available at `http://svmlight.joachims.org`.

Table 8.3. The performance of the proposed method based on different reference functions. See Table 8.1 for definitions of reference functions. The last two rows show the performance of Hist-SVM and MI-SVM for comparison. The numbers listed are the average and the standard deviation of classification accuracies over 5 random test sets. The images belong to Category 0 to Category 9. Training and test sets are of equal size.

Gaussian	$81.5\% \pm 2.2\%$
Cauchy	$81.6\% \pm 2.0\%$
Laplace	$80.6\% \pm 1.6\%$
Hyperbolic Secant	$81.8\% \pm 2.1\%$
Squared Sinc	$82.0\% \pm 2.2\%$
Symmetric Triangle	$81.7\% \pm 1.2\%$
Hist-SVM	$66.7\% \pm 1.8\%$
MI-SVM	$74.7\% \pm 0.5\%$

Section 8.3.3. The top five rows of Table 8.3 show the classification accuracies of our proposed approach with 6 different reference functions. The kernel defined by Gaussian reference function is exactly the Gaussian kernel commonly used in SVMs. It is interesting to observe that different reference functions have very similar performance. Among six reference functions, squared sinc function produces the highest average classification accuracy (82.0%). The lowest average classification accuracy is given by Laplace function (80.6%). However, the difference is not significant as indicated by the standard deviations. Therefore, for the rest experiments, we only report the results given by Gaussian reference function. One expected observation is that the proposed approach performs much better than Hist-SVM with a 14.8% (for Gaussian reference function) difference in average classification accuracy. This seems to suggest that, compared with color histograms, a region-based image representation may provide more information about a concept of image category. Another observation is that the average accuracy of the proposed method using Gaussian reference function is 6.8% higher than that of MI-SVM. As

Table 8.4. The confusion matrix of image categorization experiments (over 5 randomly generated test sets). Each row lists the average percentage of images (test images) in one category classified to each of the 10 categories by the proposed method using Gaussian reference function. Numbers on the diagonal show the classification accuracy for each category.

	Cat. 0	Cat. 1	Cat. 2	Cat. 3	Cat. 4	Cat. 5	Cat. 6	Cat. 7	Cat. 8	Cat. 9
Cat. 0	**67.7**%	3.7%	5.7%	0.0%	0.3%	8.7%	5.0%	1.3%	0.3%	7.3%
Cat. 1	1.0%	**68.4**%	4.3%	4.3%	0.0%	3.0%	1.3%	1.0%	15.0%	1.7%
Cat. 2	5.7%	5.0%	**74.3**%	2.0%	0.0%	3.3%	0.7%	0.0%	6.7%	2.3%
Cat. 3	0.3%	3.7%	1.7%	**90.3**%	0.0%	0.0%	0.0%	0.0%	1.3%	2.7%
Cat. 4	0.0%	0.0%	0.0%	0.0%	**99.7**%	0.0%	0.0%	0.0%	0.0%	0.3%
Cat. 5	5.7%	3.3%	6.3%	0.3%	0.0%	**76.0**%	0.7%	4.7%	2.3%	0.7%
Cat. 6	3.3%	0.0%	0.0%	0.0%	0.0%	1.7%	**88.3**%	2.3%	0.7%	3.7%
Cat. 7	2.3%	0.3%	0.0%	0.0%	0.0%	2.0%	1.0%	**93.4**%	0.7%	0.3%
Cat. 8	0.3%	15.7%	5.0%	1.0%	0.0%	4.3%	1.0%	0.7%	**70.3**%	1.7%
Cat. 9	3.3%	1.0%	0.0%	3.0%	0.7%	1.3%	1.0%	2.7%	0.0%	**87.0**%

we will see in Section 8.3.4, the difference becomes even greater as the number of categories increases. This suggests that the proposed method is more effective than MI-SVM in learning concepts of image categories under the same image representation. The MIL formulation of our method may be better suited for region-based image classification than that of MI-SVM.

Next, we take a closer analysis of the performance by looking at classification results on every category in terms of "confusion matrix." The results are listed in Table 8.4. Each row lists the average percentage of images in one category classified to each of the 10 categories by the proposed method using Gaussian reference function. The numbers on the diagonal show the classification accuracy for each category, and off-diagonal entries indicate classification errors. Ideally, one would expect the diagonal terms be all 1's, and the off-diagonal terms be all 0's. A detailed examination of the

Fig. 8.15. Some sample images taken from two categories: "Beach" and "Mountains and glaciers."

"confusion matrix" shows that two of the largest errors (the underlined numbers in Table 8.4) are errors between Category 1 (Beach) and Category 8 (Mountains and glaciers): 15.0% of beach images are misclassified as mountains and glaciers; 15.7% of mountains and glaciers images are misclassified as beach. Figure 8.15 presents 12 misclassified images (in at least one experiment) from both categories. All beach images in Figure 8.15 contain mountains or mountain-like regions, while all the mountains and glaciers images have regions corresponding to river, lake, or even ocean. In other words, although these two image categories do not share annotation words, they are semantically related and visually similar. This may be the reason for the relatively highest classification errors.

8.3.3 Sensitivity to Image Segmentation

Because image segmentation cannot be perfect, being robust to segmentation-related uncertainties becomes a critical performance index for a region-based image classification method. In this section, we compare the performance of the proposed

method with MI-SVM approach when the coarseness of image segmentation varies. As mentioned in Section 8.3.1, MI-SVM is also a region-based classification approach, and uses the same image representation as our proposed method. To give a fair comparison, we control the coarseness of image segmentation by adjusting the stop criteria of the k-means segmentation algorithm. We pick 5 different stop criteria. The corresponding average numbers of regions per image (computed over 1000 images from Category 0 to Category 9) are 4.31, 6.32, 8.64, 11.62, and 12.25. The average and standard deviation of classification accuracies (over 5 randomly generated test sets) under each coarseness level are presented in Figure 8.16.

The results in Figure 8.16 indicate that our method outperforms MI-SVM on all 5 coarseness levels. In addition, for our method, there are no significant changes in the average classification accuracy for different coarseness levels. While the performance of MI-SVM degrades as the average number of regions per image increases. The difference in average classification accuracies between the two methods are 6.8%, 9.5%, 11.7%, 13.8%, and 27.4% as the average number of regions per image increases. This appears to support the claim that the proposed region-based image classification method is not sensitive to image segmentation.

8.3.4 Sensitivity to the Number of Categories in a Dataset

Although the experimental results in Section 8.3.2 and 8.3.3 demonstrate the good performance of the proposed method using 1000 images in Category 0 to Category 9, the scalability of the method remains to be a question: how the performance scales as the number of categories in a dataset increases. We attempt to empirically answer

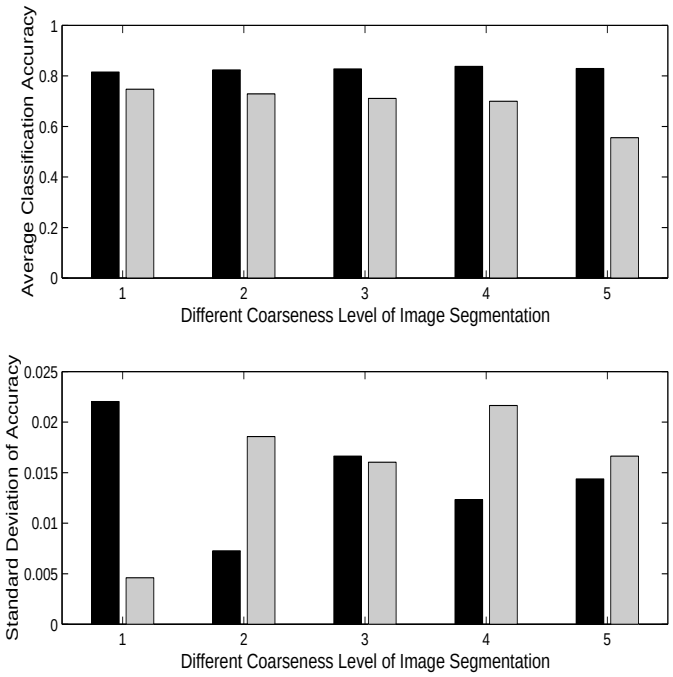


Fig. 8.16. Comparing our method with MI-SVM on the robustness to image segmentation. The experiment is performed on 1000 images in Category 0 to Category9 (training and test sets are of equal size). The top and bottom bar-plots show the average and standard deviation of classification accuracies (over 5 randomly generated test sets), respectively. There are five groups of bars in each bar-plot. From left to right, each group corresponds to a distinct stop criterion with the average number of regions per image being 4.31, 6.32, 8.64, 11.62, and 12.25, respectively. The results of our method are denoted by the bars with darker color. While the bars with lighter color represent the results for MI-SVM.

this question by performing image classification experiments over datasets with different numbers of categories. A total of 11 datasets are used in the experiments. The number of categories in a dataset varies from 10 to 20. A dataset with i categories contains $100 \times i$ images from Category 0 to Category $i - 1$. The average and standard deviation of classification accuracies (over 5 randomly generated test sets) for each dataset are presented in Figure 8.17 that includes the results of MI-SVM for comparison.

We observe a decrease in average classification accuracy as the number of categories increases. When the number of categories becomes doubled (increasing from 10 to 20 categories), the average classification accuracy of our method drops from 81.5% to 67.5%. However, our method seems to be less sensitive to the number of categories in a dataset than MI-SVM. This is indicated, in Figure 8.18, by the difference in average classification accuracies between the two methods as the number of categories in a dataset increases. It should be clear that our method outperforms MI-SVM consistently. And the performance discrepancy increases as the increase of number of categories. For the 1000-image dataset with 10 categories, the difference is 6.8%. This number is nearly doubled (12.9%) when the number of categories becomes 20. In other words, the performance degradation of our method is slower than that of MI-SVM as the number of categories increases.

8.3.5 Speed

On average, the leaning of each binary classifier using a training set of 500 images (4.31 regions per image) takes around 40 minutes of CPU time on a Pentium III 700MHz PC running Linux operation system. Among this amount of time, the majority part is

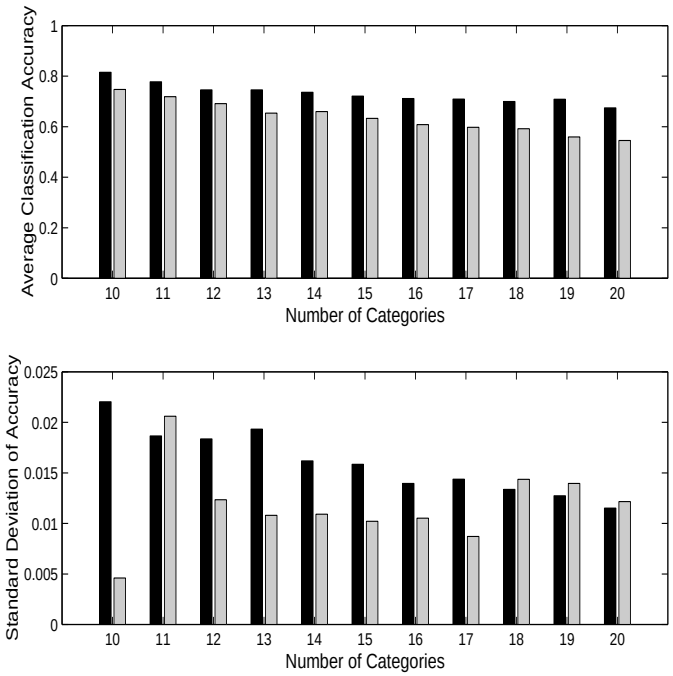


Fig. 8.17. Comparing our method with MI-SVM on the robustness to the number of categories in a dataset. The experiment is performed on 11 different datasets. The number of categories in a dataset varies from 10 to 20. A dataset with i categories contains $100 \times i$ images from Category 0 to Category $i - 1$ (training and test sets are of equal size). The top and bottom bar-plots show the average and standard deviation of classification accuracies (over 5 randomly generated test sets), respectively. The results of our method are denoted by the bars with darker color. While the bars with lighter color represent the results for MI-SVM.

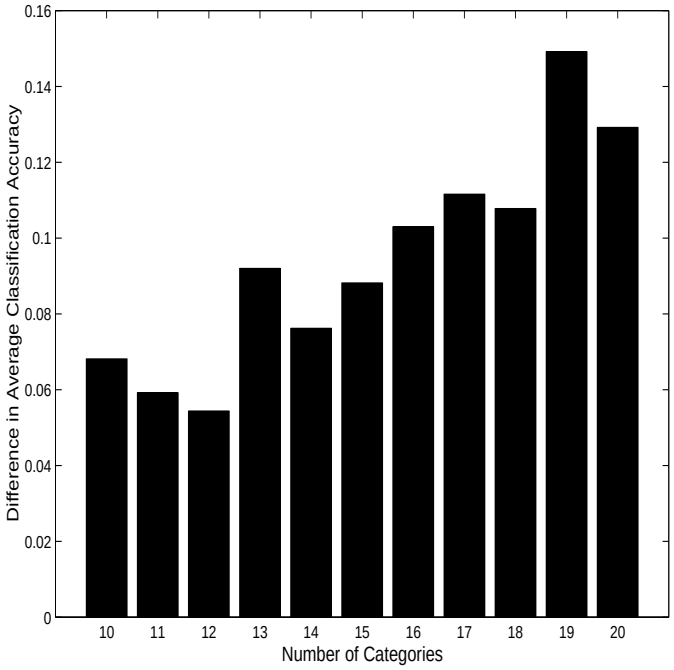


Fig. 8.18. Difference in average classification accuracies between our method and MI-SVM as the number of categories varies. A positive number indicates that our method has higher average classification accuracy.

spent on learning RPs, in particular, the `FOR` loop of **LearnPRs($\mathcal{D}$)** in Algorithm 7.1. This is because the quasi-newton search (the code is written in C programming language) needs to be applied with every instance in every positive bag as starting points (each optimization only takes a few seconds). However, since these optimizations are independent of each other, they can be fully parallelized. Thus the training time may be reduced significantly.

Chapter 9

Conclusions and Future Work

In Section 9.1, we summarize the major contributions of the thesis. The limitations of the proposed approaches are discussed in Section 9.2. Suggestions for future work are presented in Section 9.3.

9.1 Summary

A major difficulty in CBIR is the "semantic gap." It reflects the discrepancy between low-level visual features and high-level concepts. With the ultimate goal of narrowing the semantic gap, this thesis makes three contributions to the field of CBIR.

The first contribution is UFM (Chapter 5), a robust image similarity measure using fuzzified region features. In the UFM scheme, an image is first segmented into regions. Each region is then represented by a fuzzy feature that is determined by center location (a feature vector) and width (grade of fuzziness). Compared with the conventional region representation using a single feature vector, each region is represented by a set of feature vectors each with a value denoting its degree of membership to the region. Consequently, the membership functions of fuzzy sets naturally characterize the gradual transition between regions within an image. That is, they characterize the blurring boundaries due to imprecise segmentation.

A direct consequence of fuzzy feature representation is the region-level similarity. Instead of using the Euclidean distance between two feature vectors, a fuzzy similarity measure, which is defined as the maximum value of the membership function of the intersection of two fuzzy features, is used to describe the resemblance of two regions. This value is always within $[0, 1]$ with a larger value indicating a higher degree of similarity between two fuzzy features. The value depends on both the Euclidean distance between the center locations and the grades of fuzziness of two fuzzy features. Intuitively, even though two fuzzy features are close to each other, if they are not "fuzzy" (i.e., the boundary between two regions is distinctive), then their similarity could be low. In the case that two fuzzy features are far away from each other, but they are very "fuzzy" (i.e., the boundary between two regions is very blurring), the similarity could be high. These correspond reasonably to the viewpoint of the human perception.

Trying to provide a comprehensive and robust "view" of similarity between images, the region-level similarities are combined into an image-level similarity vector pair, and then the entries of the similarity vectors are weighted and added up to produce the UFM similarity measure which depicts the overall resemblance of images in color, texture, and shape properties. The comprehensiveness and robustness of UFM measure can be examined from two perspectives namely the contents of similarity vectors and the way of combining them. Each entry of similarity vectors signifies the degree of closeness between a fuzzy feature in one image and all fuzzy features in the other image. Intuitively, an entry expresses how similar a region of one image is to all regions of the other image. Thus a region is allowed to be matched with several regions in case of inaccurate image segmentation which in practice occurs quite often. By weighted summation, every

fuzzy feature in both images contributes a portion to the overall similarity measure. This further reduces the sensitivity of UFM measure. The application of the UFM method to a database of about 60,000 general-purpose images has demonstrated good accuracy and excellent robustness to image segmentation and image alterations.

The second contribution is CLUE (Chapter 6), an image retrieval scheme using unsupervised learning. CLUE attempts to retrieve semantically coherent image clusters, instead of a set of images ranked by a similarity measure. Although the underlying image semantics structure of a large image database may be quite complex, CLUE makes a rather simple assumption: semantically similar images tend to be clustered. Clustering is performed in a query-dependent way: query image and target images, which are in the neighborhood of the query in terms of a similarity measure, are clustered. As a result, CLUE generates clusters that are tailored to characteristics of the query image. CLUE employs a graph representation of images: images are viewed as nodes and similarities between images are denoted by weights of the edges connecting nodes. The graph representation captures the pairwise relationship between images, and enables CLUE to handle the metric and nonmetric similarity measures in a uniform way. In this sense, CLUE is a general approach that can be combined with any real-valued symmetric image similarity measure, and thus, may be embedded in many current CBIR systems.

Under a graph representation, clustering is naturally formulated as a graph partitioning problem. The Ncut technique is used by CLUE. The resulting image clusters are organized in a linear order specified by the traversal of the tree generated by recursive Ncut. A representative image is also found for each cluster. The system presents the clusters and the images inside to a user via a two-level display scheme. The application

of CLUE (with UFM similarity measure) to a database of $60,000$ general-purpose images demonstrates that CLUE can indeed provide more semantic clues to a system user than an existing CBIR system using the same similarity measure. Numerical evaluations on a 1000-image database show good cluster quality and improved retrieval accuracy. Furthermore, preliminary results on images returned by Google's Image Search suggest the potential of applying CLUE to real world image data and integrating CLUE as a part of the interface for keyword-based image retrieval systems.

The last contribution is an image categorization method that classifies images based on the information of regions (Chapter 4 and Chapter 7). Each image is represented as a collection of regions obtained from image segmentation using k-means algorithm. The classification is guided by a set of automatically derived rules that relate the concept underlying an image category with the occurrence of regions (of certain types) in an image. To incorporate the uncertainties that are intrinsic to image segmentation, each rule is modeled as a fuzzy inference rule. And the classifier built upon such rules becomes a fuzzy rule-based classifier. In Chapter 4 we prove that, under quite general conditions, the proposed classifier is functionally equivalent to SVMs with a special class of kernels. Therefore, SVM learning is applied to train such classifiers. In particular, each rule is determined by a support vector and the associated Lagrange multiplier. We demonstrate that the proposed method performs well in classifying images from 20 semantic classes.

9.2 Limitations

A major limitation of the UFM scheme, which is inherent to the current fuzzy feature representation, is that the specificity is sacrificed to the robustness. The current system works well for the testing image database that consists of 60,000 photographic pictures. However, experiments on a different image database (also available at the demonstration web site) of about 140,000 clip art pictures show that the IRM outperforms the UFM a little in accuracy. This is because, unlike photographs, segmentation of a clip art picture tends to be very accurate. Fuzzy features blur the boundaries of the originally clear-cut regions, which makes accurately recognizing and matching similar regions even harder.

CLUE also has several limitations:

- The current heuristic used in the recursive Ncut always bipartitions the largest cluster. This is a low-complexity rule and is computationally efficient to implement. But it may divide a large and pure cluster into several clusters even when there exists a smaller and semantically more diverse cluster. Bipartitioning the semantically most diverse cluster seems to be more reasonable. However, an open question is how to automatically and efficiently estimate the semantic diversity of a cluster.

- The current method of finding a representative image for a cluster does not always give a semantically representative image. For the example in Figure 8.8(a), one would expect the representative image to an image of a bird. But the system

chooses an image of sheep (the third image). This discrepancy is due to the semantic gap: an image that is most similar to all images in the cluster in terms of a similarity measure does not necessarily belong to the dominant semantic class of the cluster.

- If the number of neighboring target images is large (more than several thousand), sparsity of the affinity matrix becomes crucial to retrieval speed. The current weighting scheme given by (6.1) does not lead to a sparse affinity matrix. As a result, different weighting schemes should be studied to improve the scalability of CLUE.

For the proposed image categorization algorithm, the definition of DD function may be improved. The current definition of DD function, which is a multiplicative model, is very sensitive to instances in negative bags. It can be easily observed from (7.1) that the DD value at a point is significantly reduced if there is a single instance from negative bags close to the point. This property may be desirable for some applications, such as drug discovery [68], where the goal is to learn a single point in the instance feature space with the maximum DD value from an almost "noise free" dataset. But this is not a typical problem setting for region-based image categorization where data usually contain noise. Thus a more robust definition of DD, such as an additive model, is likely to enhance the performance.

9.3 Future Work

In future work, we intend to pursue in the following areas:

- Feature selection

  One of the advantages of region-based image retrieval methods is that the size, shape, and absolute and relative location of the regions can provide additional help. But in the current image segmentation, location information is not fully exploited. We plan to test other segmentation algorithms, such as the one described in [32], which include the location information in the segmentation process.

- Learning techniques

  One possible future direction is to integrate CLUE with keyword-based image retrieval approaches. Other graph theoretic clustering techniques [70] need to be tested for possible performance improvement. CLUE may be combined with nonlinear dimensionality reduction techniques, such as the methods in [86] and [108], to provide a global visualization together with a local retrieval. The current RP learning scheme may be combined with boosting technique.

- Applications

  We are planning to apply the proposed algorithms to special image databases including digital imagery for art and cultural heritages, and biomedical images. In terms of the size of images and the level of details required in image representation, these applications are more challenging than the experiments, on which the proposed algorithms have been tested.

References

[1] S. Abe and R. Thawonmas, "A Fuzzy Classifier with Ellipsoidal Regions," *IEEE Transactions on Fuzzy Systems*, vol. 5, no. 3, pp. 358-368, 1997.

[2] S. Andrews, I. Tsochantaridis, and T. Hofmann, "Support Vector Machines for Multiple-Instance Learning," *Advances in Neural Information Processing Systems* 15, Cambridge, MA: MIT Press, 2003.

[3] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval,* Addison-Wesley, 1999.

[4] H. Bandemer and W. Nather, *Fuzzy Data Analysis*, Kluwer Academic Publishers, 1992.

[5] K. Barnard and D. Forsyth, "Learning the Semantics of Words and Pictures," *Proc. 8th Int. Conference on Computer Vision*, vol. 2, pp. 408–415, 2001.

[6] P. L. Bartlett, "For Valid Generalization, the Size of the Weights is More Important Than the Size of the Network," in *Advances in Neural Information Processing Systems 9*, M.C. Mozer, M.I. Jordan, and T. Petsche, (eds.), Cambridge, MA: The MIT Press, pp. 134-140, 1997.

[7] A. Del Bimbo and P. Pala, "Visual Image Retrieval by Elastic Matching of User Sketches," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 2, pp. 121–132, 1997.

[8] P. S. Bradley and O. L. Mangasarian, "Feature Selection via Concave Minimization and Support Vector Machines," *Proceedings of the 15th International Conference on Machine Learning*, pp. 82-90, Morgan Kaufmann, San Francisco, CA, 1998.

[9] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121-167, 1998.

[10] C. Carson, S. Belongie, H. Greenspan, and J. Malik, "Blobworld: Image Segmentation Using Expectation-Maximization and its Application to Image Querying," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 8, pp. 1026–1038, 2002.

[11] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," [http://www.csie.ntu.edu.tw/∼cjlin/libsvm], 2001.

[12] S.-K. Chang, Q.-Y. Shi, and C.-W. Yan, "Iconic Indexing by 2D Strings," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 9, no. 3, pp. 413–428, 1987.

[13] O. Chapelle, P. Haffner, and V. N. Vapnik, "Support Vector Machines for Histogram-Based Image Classification," *IEEE Trans. Neural Networks*, vol. 10, no. 5, pp. 1055–1064, 1999.

[14] S.-M. Chen, Y.-J. Horng, and C.-H. Lee, "Document Retrieval Using Fuzzy-Valued Concept Networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 31, no. 1, pp. 111-118, 2001.

[15] Y. Chen, J. Z. Wang, and J. Li, "FIRM: Fuzzily Integrated Region Matching for Content-Based Image Retrieval," *Proc. ACM Multimedia*, pp. 543–545, 2001.

[16] Y. Chen and J. Z. Wang, "A Region-Based Fuzzy Feature Matching Approach to Content-Based Image Retrieval," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 9, pp. 1252–1267, 2002.

[17] Y. Chen and J. Z. Wang, "Support Vector Learning for Fuzzy Rule-Based Classification Systems," *IEEE Trans. Fuzzy Systems*, vol. 11, 2003. (To appear)

[18] Y. Chen, J. Z. Wang, and R. Krovetz, "An Unsupervised Learning Approach to Content-Based Image Retrieval," *Proc. IEEE Int'l Symposium on Signal Processing and its Applications*, 2003.

[19] Y. Chen and J. Z. Wang, "Kernel Machines and Additive Fuzzy Systems: Classification and Function Approximation," *Proc. IEEE Int'l Conference on Fuzzy Systems*, pp. 789–795, 2003.

[20] Y. Chen and J. Z. Wang, "Looking Beyond Region Boundaries: A Robust Image Similarity Measure Using Fuzzified Region Features," *Proc. IEEE Int'l Conference on Fuzzy Systems*, pp. 1165–1170, 2003.

[21] C.-K. Chiang, H.-Y, Chung, and J.-J Lin, "A Self-Learning Fuzzy Logic Controller Using Genetic Algorithms with Reinforcements," *IEEE Transactions on Fuzzy Systems*, vol. 5, no. 3, pp. 460–467, 1997.

[22] J. Costeira and T. Kanade, "A Multibody Factorization Method for Motion Analysis," *Proc. Int'l Conf. Computer Vision*, pp. 1071–1076, 1995.

[23] I. J. Cox, M. L. Miller, T. P. Minka, T. V. Papathomas, and P. N. Yianilos, "The Bayesian Image Retrieval System, PicHunter: Theory, Implementation, and Psychophysical Experiments," *IEEE Trans. Image Processing*, vol. 9, no. 1, pp. 20–37, 2000.

[24] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, 2000.

[25] I. Daubechies, *Ten Lectures on Wavelets*, Capital City Press, 1992.

[26] J. A. Dickerson and B. Kosko, "Fuzzy Function Approximation with Ellipsoidal Rules," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 4, pp. 542-560, 1996.

[27] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Perez, "Solving the Multiple Instance Problem with Axis-Parallel Rectangles," *Artificial Intelligence*, vol. 89, no. 1-2, pp. 31–71, 1997.

[28] D. Dubois and H. Prade, "Operations on Fuzzy Numbers," *International Journal of Systems Science*, vol. 9, no. 6, pp. 613-626, 1978.

[29] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, Second Edition, John Wiley and Sons, Inc., 2000.

[30] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz, "Efficient and Effective Querying by Image Content," *J. Intell. Inform. Syst.*, vol. 3, no. 3-4, pp. 231–262, 1994.

[31] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, Prentice Hall, 2002.

[32] H. Frigui and S. Salem, "Fuzzy Clustering and Subset Feature Weighting," *Proc. IEEE Int'l Conf. on Fuzzy Systems*, pp. 857–862, 2003.

[33] Y. Gdalyahu and D. Weinshall, "Flexible Syntactic Matching of Curves and Its Application to Automatic Hierarchical Classification of Silhouettes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 21, no. 12, pp. 1312–1328, 1999.

[34] Y. Gdalyahu, D. Weinshall, and M. Werman, "Self-Organization in Vision: Stochastic Clustering for Image Segmentation, Perceptual Grouping, and Image Database Organization," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23, no. 10, pp. 1053–1074, 2001.

[35] S. Geman, E. Bienenstock, and R. Doursat, "Neural Networks and the Bias/Variance Dilemma," *Neural Computation*, vol. 4, no. 1, pp. 1-58, 1992.

[36] M. G. Genton, "Classes of Kernels for Machine Learning: A Statistics Perspective," *Journal of Machine Learning Research*, vol. 2, pp. 299-312, 2001.

[37] A. Gersho, "Asymptotically Optimum Block Quantization," *IEEE Trans. Information Theory*, vol. 25, no. 4, pp. 373–380, 1979.

[38] T. Gevers and A. W. M. Smeulders, "PicToSeek: Combining Color and Shape Invariant Features for Image Retrieval," *IEEE Trans. Image Processing*, vol. 9, no. 1, pp. 102–119, 2000.

[39] G. H. Golub and C. F. Van Loan, *Matrix Computations,* 3rd ed., Johns Hopkins University Press, 1996.

[40] M. M. Gorkani and R. W. Picard, "Texture Orientation for Sorting Photos 'at a glance'," *Proc. 12th Int'l Conf. on Pattern Recognition*, vol. I, pp. 459–464, 1994.

[41] A. Gupta and R. Jain, "Visual Information Retrieval," *Commun. ACM*, vol. 40, no. 5, pp. 70–79, 1997.

[42] J. Hafner, H. S. Sawhney, W. Equitz, M. Flickner, and W. Niblack, "Efficient Color Histogram Indexing for Quadratic Form Distance Functions," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, no. 7, pp. 729–736, 1995.

[43] J. A. Hartigan and M. A. Wong, "Algorithm AS136: A k-means Clustering Algorithm," *Applied Statistics*, vol. 28, pp. 100–108, 1979.

[44] R. J. Hathaway and J. C. Bezdek, "Fuzzy c-means Clustering of Incomplete Data," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 31, no. 5, pp. 735-744, 2001.

[45] M. A. Hearst and J. O. Pedersen, "Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results," *Proc. of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)*, pp. 76–84, 1996.

[46] F. Hoppner, F. Klawonn, R. Kruse, and T. Runkler, *Fuzzy Cluster Analysis: Methods For Classification, Data Analysis and Image Recognition*, John Wiley & Sons, LTD, 1999.

[47] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 1985.

[48] J. Huang, S. R. Kumar, and R. Zabih, "An Automatic Hierarchical Image Classification Scheme," *Proc. 6th ACM Int'l Conf. on Multimedia*, pp. 219–228, 1998.

[49] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing Images Using the Hausdorff Distance," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, no. 9, pp. 850–863, 1993.

[50] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Construction of Fuzzy Classification Systems with Rectangular Fuzzy Rules Using Genetic Algorithms," *Fuzzy Sets and Systems*, vol. 65, pp. 237-253, 1994.

[51] D. W. Jacobs, D. Weinshall, and Y. Gdalyahu, "Classification with Nonmetric Distances: Image Retrieval and Class Representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 6, pp. 583–600, 2000.

[52] L. Jia and L. Kitchen, "Object-Based Image Similarity Computation Using Inductive Learning of Contour-Segment Relations," *IEEE Trans. Image Processing*, vol. 9, no. 1, pp. 80–87, 2000.

[53] J.-S. R. Jang and C. T. Sun, "Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems," *IEEE Transactions on Neural Networks*, vol. 4, no. 1, pp. 156-159, 1993.

[54] J.-S. R. Jang and C.-T. Sun, "Neuro-Fuzzy Modeling and Control," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 378-406, 1995.

[55] T. Joachims, "Making Large-Scale SVM Learning Practical," *Advances in Kernel Methods - Support Vector Learning*, edited by B. Schölkopf, C. J.C. Burges, and A.J. Smola, Cambridge, MA: MIT Press, pp. 169-184, 1999.

[56] L. Kaufman, "Solving the Quadratic Programming Problem Arising in Support Vector Classification," *Advances in Kernel Methods - Support Vector Learning*, edited by B. Schölkopf, C. J.C. Burges, and A.J. Smola, Cambridge, MA: MIT Press, pp. 147-167, 1999.

[57] F. Klawon and P. E. Klement, "Mathematical Analysis of Fuzzy Classifiers," in *Lecture Notes in Computer Science*, vol. 1280, pp. 359-370, 1997.

[58] G. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice Hall, 1995.

[59] A. Kontanzad and Y. H. Hong, "Invariant Image Recognition by Zernike Moments," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, no. 5, pp. 489–497, 1990.

[60] B. Kosko, *Fuzzy Engineering*, Prentice Hall, 1996.

[61] S. Kulkarni, B. Verma, P. Sharma, and H. Selvaraj, "Content Based Image Retrieval Using a Neuro-Fuzzy Technique," *Proc. IEEE Int'l Joint Conf. on Neural Networks*, pp. 846–850, July 1999.

[62] C. C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part I, Part II," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 2, pp. 404-435, 1990.

[63] J. Li and J. Z. Wang, "Automatic Linguistic Indexing of Pictures By a Statistical Modeling Approach," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 10, 2003.

[64] J. Li, J. Z. Wang, and G. Wiederhold, "IRM: Integrated Region Matching for Image Retrieval," *Proc. 8th ACM Int'l Conf. on Multimedia*, pp. 147–156, 2000.

[65] J. Li, J. Z. Wang, and G. Wiederhold, "Classification of Textured and Non-Textured Images Using Region Segmentation," *Proc. 7th Int'l Conf. on Image Processing*, pp. 754–757, September 2000.

[66] W. Y. Ma and B. Manjunath, "NeTra: A Toolbox for Navigating Large Image Databases," *Proc. IEEE Int'l Conf. Image Processing*, pp. 568–571, 1997.

[67] O. Maron and A. L. Ratan, "Multiple-Instance Learning for Natural Scene Classification," *Proc. 15th Int'l Conf. on Machine Learning*, pp. 341–349, 1998.

[68] O. Maron and T. Lozano-Pérez, "A Framework for Multiple-Instance Learning," *Advances in Neural Information Processing Systems* 10, Cambridge, MA: MIT Press, pp. 570–576, 1998.

[69] D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, W H Freeman & Co., 1983.

[70] D. W. Matula, "Graph Theoretic Techniques for Cluster Analysis Algorithm," *Classification and Clustering*, Ed., J. Van Ryzin, New York: Academic Press, pp. 95–129, 1977.

[71] S. Mehrotra, Y. Rui, M. Ortega-Binderberger, and T. S. Huang, "Supporting Content-Based Queries over Images in MARS," *Proc. IEEE Int'l Conf. on Multimedia Computing and Systems*, pp. 632–633, June 1997.

[72] J. Mercer, "Functions of Positive and Negative Type and Their Connection with the Theory of Integral Equations," *Philosophical Transactions of the Royal Society London*, A209, pp. 415-446, 1909.

[73] T. P. Minka and R. W. Picard, "Interactive Learning with a 'Society of Models'," *Pattern Recognition*, vol. 30, no. 4, pp. 565–581, 1997.

[74] S. Mitaim and B. Kosko, "The Shape of Fuzzy Sets in Adaptive Function Approximation," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 4, pp. 637-656, 2001.

[75] S. Miyamoto, "Two Approaches for Information Retrieval Through Fuzzy Associations," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 1, pp. 123-130, 1989.

[76] A. Mojsilovic, J. Kovacevic, J. Hu, R. J. Safranek, and S. K. Ganapathy, "Matching and Retrieval Based on the Vocabulary and Grammar of Color Patterns," *IEEE Trans. Image Processing*, vol. 9, no. 1, pp. 38–54, 2000.

[77] V. Ogle and M. Stonebraker, "Chabot: Retrieval from a Relational Database of Images," *IEEE Computer*, vol. 28, no. 9, pp. 40–48, 1995.

[78] P. J. Pacini and B. Kosko, "Adaptive Fuzzy Frequency Hopper," *IEEE Transactions on Communications*, vol. 43, no. 6, pp. 2111-2117, 1995.

[79] A. Pentland, R. W. Picard, and S. Sclaroff, "Photobook: Content-Based Manipulation for Image Databases," *Int'l J. Comput. Vis.*, vol. 18, no. 3, pp. 233–254, 1996.

[80] P. Perona and W. Freeman, "A Factorization Approach to Grouping," *Proc. European Conf. Computer Vision*, pp. 655–670, 1998.

[81] R. W. Picard and T. P. Minka, "Vision Texture for Annotation," *Journal of Multimedia Systems*, vol.3, no. 1, pp. 3–14, 1995.

[82] J. C. Platt, "Fast Training of Support Vector Machines Using Sequential Minimal Optimization," *Advances in Kernel Methods - Support Vector Learning*, edited by B. Schölkopf, C. J.C. Burges, and A.J. Smola, Cambridge, MA: MIT Press, pp. 185-208, 1999.

[83] A. Pothen, H. D. Simon, and K. P. Liou, "Partitioning Sparse Matrices with Eigenvectors of Graphs," *SIAM J. Matrix Analytical Applications*, vol. 11, pp. 430–452, 1990.

[84] S. A. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: the art of scientific computing*, second edition, Cambridge University Press, New York, 1992.

[85] R. Rovatti, "Fuzzy Piecewise Multilinear and Piecewise Linear Systems as Universal Approximators in Sobolev Norms," *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 2, pp. 235-249, 1998.

[86] S. T. Roweis and L. K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, vol. 290, pp. 2323–2326, 2000.

[87] Y. Rubner, L. J. Guibas, and C. Tomasi, "The Earth Mover's Distance, Multi-Dimensional Scaling, and Color-Based Image Retrieval," *Proc. DARPA Image Understanding Workshop*, pp. 661–668, May 1997.

[88] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra, "Relevance Feedback: A Power Tool for Interactive Content-Based Image Retrieval," *IEEE Trans. Circuits and Video Technology*, vol. 8, no. 5, pp. 644–655, 1998.

[89] S. Santini and R. Jain, "Similarity Measures," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 21, no. 9, pp. 871–883, 1999.

[90] S. Sarkar and P. Soundararajan, "Supervised Learning of Large Perceptual Organization: Graph Spectral Partitioning and Learning Automata," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 5, pp. 504–525, 2000.

[91] F. Sattar and D. B. H. Tay "Enhancement of Document Images Using Multiresolution and Fuzzy Logic Techniques," *IEEE Signal Processing Letters*, vol. 6, no. 10, pp. 249-252, 1999.

[92] B. Schölkopf, K.-K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik, "Comparing support vector machines with Gaussian kernels to radial basis function classifiers," *IEEE Transactions on Signal Processing*, vol.45, no. 11, pp. 2758-2765, 1997.

[93] C. Schmid and R. Mohr, "Local Grayvalue Invariants for Image Retrieval," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 5, pp. 530–535, 1997.

[94] M. Setnes, "Supervised Fuzzy Clustering for Rule Extraction," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 4, pp. 416-424, 2000.

[95] G. Sheikholeslami, W. Chang, and A. Zhang, "SemQuery: Semantic Clustering and Querying on Heterogeneous Features for Visual Data," *IEEE Trans. Knowledge and Data Engineering*, vol. 14, no. 5, pp. 988–1002, 2002.

[96] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

[97] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-Based Image Retrieval at the End of the Early Years," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 12, pp. 1349–1380, 2000.

[98] J. R. Smith and S.-F. Chang, "VisualSEEK: A Fully Automated Content-Based Query System," *Proc. 4th ACM Int'l Conf. on Multimedia*, pp. 87–98, 1996.

[99] J. R. Smith and C.-S. Li, "Image Classification and Querying Using Composite Region Templates," *Int'l J. Computer Vision and Image Understanding*, vol. 75, nos. 1/2, pp. 165–174, 1999.

[100] A. J. Smola, B. Schölkopf, and K.-R. Müller, "The Connection Between Regularization Operators and Support Vector Kernels," *Neural Networks*, vol. 11, no. 4, pp. 637-649, 1998.

[101] T. M. Strat, *Natural Object Recognition*, Berlin: Springer-Verlag, 1992.

[102] M. Sugeno and G. T. Kang, "Structure Identification of Fuzzy Model," *Fuzzy Sets and Systems*, vol. 28, pp. 15-33, 1988.

[103] Y. Suzuki, K. Itakura, S. Saga, and J. Maeda, "Signal Processing and Pattern Recognition with Soft Computing," *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1297-1317, 2001.

[104] M. J. Swain and B. H. Ballard, "Color Indexing," *Int'l J. Comput. Vis.*, vol. 7, no. 1, pp. 11–32, 1991.

[105] D. L. Swets and J. Weng, "Using Discriminant Eigenfeatures for Image Retrieval," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, no. 8, pp. 831–837, 1996.

[106] M. Szummer and R. W. Picard, "Indoor-Outdoor Image Classification," *Proc. IEEE Int'l Workshop on Content-Based Access of Image and Video Databases*, pp. 42–51, 1998.

[107] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and Its Applications to Modeling and Control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, no. 1, pp. 116-132, 1985.

[108] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, pp. 2319–2323, 2000.

[109] R. Thawonmas and S. Abe, "Function Approximation Based on Fuzzy Rules Extracted From Partitioned Numerical Data," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 29, no. 4, pp. 525-534, 1999.

[110] S. Tong and E. Chang, "Support Vector Machine Active Learning for Image Retrieval," *Proc. 9th ACM Int'l Conf. on Multimedia*, pp. 107–118, 2001.

[111] M. Unser, "Texture Classification and Segmentation Using Wavelet Frames," *IEEE Trans. Image Processing*, vol. 4, no. 11, pp. 1549–1560, 1995.

[112] A. Vailaya, M. A. T. Figueiredo, A. K. Jain, and H.-J. Zhang, "Image Classification for Content-Based Indexing," *IEEE Trans. Image Processing*, vol. 10, no. 1, pp. 117–130, 2001.

[113] V. Vapnik, *Estimation of Dependences Based on Empirical Data (in Russian)*, Nauka, Moscow, 1979. (English translation: Springer Verlag, New York, 1982).

[114] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.

[115] V. Vapnik, *Statistical Learning Theory*, John Wiley and Sons, Inc., New York, 1998.

[116] V. Vapnik and A. Chervonenkis, "On the Uniform Convergence of Relative Frequencies of Events to Theirs Probabilities," *Theory of Probability and its Applications*, vol. 16, no. 2, pp. 264-280, 1971.

[117] V. Vapnik, S. E. Golowich, and A. Smola, "Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing," in *Advances in Neural Information Processing Systems 9*, M.C. Mozer, M.I. Jordan, and T. Petsche, (eds.), Cambridge, MA: The MIT Press, pp. 281-287, 1997.

[118] C. Vertan and N. Boujemaa, "Embedding Fuzzy Logic in Content Based Image Retrieval," *Proc. 19th Int'l Meeting of the North American Fuzzy Information Processing Society NAFIPS 2000*, pp. 85–89, July 2000.

[119] J. Z. Wang, J. Li, R. M. Gray, and G. Wiederhold, "Unsupervised Multiresolution Segmentation for Images with Low Depth of Field," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23, no. 1, pp. 85–91, 2001.

[120] J. Z. Wang, J. Li, and G. Wiederhold, "SIMPLIcity: Semantics-Sensitive Integrated Matching for Picture LIbraries," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23, no. 9, pp. 947–963, 2001.

[121] J. Z. Wang, G. Wiederhold, O. Firschein, and X. W. Sha, "Content-Based Image Indexing and Searching Using Daubechies' wavelets," *Int'l J. Digital Libraries*, vol. 1, no. 4, pp. 311–328, 1998.

[122] L.-X. Wang, *Adaptive Fuzzy Systems And Control: Design and Stability Analysis*, Englewood Cliffs, NJ: Prentice-Hall, 1994.

[123] Y. Weiss, "Segmentation Using Eigenvectors: a Unifying View," *Proc. Int'l Conf. Computer Vision*, pp. 975–982, 1999.

[124] P. Wolfe, "A Duality Theorem for Nonlinear Programming," *Quarterly of Applied Mathematics*, vol. 19, no. 3, pp. 239-244, 1961.

[125] J. Yen, "Fuzzy Logic—A Modern Perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 1, pp. 153-165, 1999.

[126] J. Yen and L. Wang, "Application of Statistical Information Criteria for Optimal Fuzzy Model Construction," *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 3, pp. 362-372, 1998.

[127] H. Ying, "General SISO Takagi-Sugeno Fuzzy Systems with Linear Rule Consequent are Universal Approximators," *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 4, pp. 582-587, 1998.

[128] H. Yu and W. Wolf, "Scenic Classification Methods for Image and Video Databases," *Proc. SPIE Int'l Conf. on Digital Image Storage and Archiving Systems*, vol. 2606, pp. 363–371, 1995.

[129] L. A. Zadeh, "Fuzzy Sets," *Information and Control*, vol. 8, pp. 338-353, 1965.

[130] Q. Zhang, S. A. Goldman, W. Yu, and J. Fritts, "Content-Based Image Retrieval Using Multiple-Instance Learning," *Proc. 19th Int'l Conf. on Machine Learning*, pp. 682–689, 2002.

[131] Q. Zhang and S. A. Goldman, "EM-DD: An Improved Multiple-Instance Learning Technique," *Advances in Neural Information Processing Systems* 14, Cambridge, MA: MIT Press, pp. 1073–1080, 2002.

[132] X. S. Zhou and T. S. Huang, "Comparing Discriminating Transformations and SVM for Learning during Multimedia Retrieval," *Proc. 9th ACM Int'l Conf. on Multimedia*, pp. 137–146, 2001.

[133] S. C. Zhu and A. Yuille, "Region Competition: Unifying Snakes, Region Growing, and Bayes/MDL for Multiband Image Segmentation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, no. 9, pp. 884–900, 1996.

[134] H.-J. Zimmermann, *Fuzzy Set Theory and Its Applications*, Kluwer Academic Publishers, 1991.

Vita

Yixin Chen received the B.S. degree from the Department of Automation, Beijing Polytechnic University, China, in 1995, the M.S. degree in control theory and application from Tsinghua University, China, in 1998, and the M.S. and Ph.D. degrees in electrical engineering from the University of Wyoming, Laramie, WY, in 1999 and 2001, respectively. Since August 2000, he has been a Ph.D student in the Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA. His research interests include machine learning, content-based image retrieval, computer vision, precision and fault tolerant robotic control, and soft computing. He is a student member of the Association for Computing Machinery (ACM), the Institute of Electrical and Electronics Engineers (IEEE), the IEEE Computer Society, the IEEE Neural Networks Society, and the IEEE Robotics and Automation Society.