Dipartimento di Informatica e
Scienze dell'Informazione

Automatic Image Annotation based on Learning Visual Cues

by

Laura Lo Gerfo

Theses Series **DISI-TH-2009-01**

DISI, Università di Genova

v. Dodecaneso 35, 16146 Genova, Italy `http://www.disi.unige.it/`

Università degli Studi di Genova

Dipartimento di Informatica e
Scienze dell'Informazione

Dottorato di Ricerca in Informatica

Ph.D. Thesis in Computer Science

Automatic Image Annotation based on Learning Visual Cues

by

Laura Lo Gerfo

February, 2009

Dottorato di Ricerca in Informatica
Dipartimento di Informatica e Scienze dell'Informazione
Università degli Studi di Genova

DISI, Univ. di Genova
via Dodecaneso 35
I-16146 Genova, Italy
http://www.disi.unige.it/

Ph.D. Thesis in Computer Science (S.S.D. INF/01)

Submitted by Laura Lo Gerfo
Dipartimento di Informatica e Scienze dell'Informazione
Università degli Studi di Genova
logerfo@disi.unige.it

Date of submission: February 2009

Title: Automatic Image Annotation Based on
Learning Visual Cues

Advisor: Alessandro Verri
Dipartimento di Informatica e Scienze dell'Informazione
Università degli Studi di Genova
verri@disi.unige.it

Ext. Reviewers:

Massimo Ferri
Dipartimento di Matematica
Università degli Studi di Bologna
ferri@dm.unibo.it

Roberto Manduchi
Department of Computer Engineering
University of California, Santa Cruz
manduchi@soe.ucsc.edu

Abstract

Efficient access to digital images requires the development of techniques to search and organize the visual information. While current technology provides several search engines relying on textual description, the research on content-based image retrieval systems faces much more challenging problems.

Traditional databases exploit manual annotation for indexing and then retrieving the proper image collections. Although manual annotation of image content is considered a "best case" in terms of accuracy, it is an expensive and time-consuming process. As opposed to manual annotation, automatic annotation in large collections of data must deal with difficult issues. First, a "broad domain" of images has a virtually unlimited and unpredictable variability in appearance even for the same semantic meaning. Another crucial point is that the user interprets an image identifying its semantic meaning by using a large amount of background and context knowledge. An automatic annotation system instead is only able to quantify and provide measurements by data processing and lacks the ability to infer information from the context.

This thesis explores an automatic strategy for semantically annotating images of a large dataset. In the context of statistical learning, automatic annotation and retrieval can be cast as classification problems where each class is defined as a group of image regions labeled with a common semantic keyword. The proposed framework is based on region-level analysis that is a good compromise between local and global approaches. We use an unsupervised learning strategy to organize the data in homogeneous clusters. In order to establish a connection between the natural language and the region descriptors we assign tags to some clusters and apply a supervised algorithm on each cluster. We then employ an architecture of classifiers able to automatically assign a set of labels to a given image and to retrieve a subset of representative images belonging to a specific semantic class.

The main contribution of this work is an effective architecture that could be expanded easily to add new semantic concepts. Extensive experiments of the proposed approach are ongoing on large databases of natural and outdoor images, commonly used to test context-based retrieval systems. The experimental results obtained so far confirm the potential of the proposed approach.

To Anna and Sofia,
images of my past and my future

I never read, I just looked at pictures.

(Andy Warhol)

Table of Contents

Chapter 1

Introduction

1.1 Motivations and background

The focus of this thesis is on the automatic annotation of natural images in large datasets or heterogenous collections. Examples of relevant applications of the work described in the following are the organization and indexing of huge web-based image repositories such as Flickr or Picasa, which are expected to keep growing rapidly thanks to the continuous insertion of new data from the users.
The *correct* annotation of images is indeed a challenging problem even for human beings because it is not trivial – if not impossible at all – to assess objectively what *correct* means in this context. Annotations may be based on different criteria such as color, texture, shape, size or other kinds of both semantic and spatial constraints. The specific balance among all such criteria depends on the user's viewpoint and the specific class of images: this makes the design and development of algorithmic strategies to image annotation an open research problem within the context of Content-based Image Retrieval (CBIR) and, more in general, of Computer Vision and Image Analysis.
Actually, it is widely acknowledged among computer vision researchers that the achievement of proper solutions to this problem is cumbersome and requires interdisciplinary efforts. Therefore, it is not plausible for any single research program to lead to a definitive automatic annotation system. Nonetheless, a number of crucial subproblems may be addressed effectively, and the complete desired system is likely to be based on future integrations of all the resulting submodules.

In the above context, the first general objective of our research work was to assess to what extent unsupervised and supervised learning may contribute to successfully annotate natural images. Consequently, in order to make the first steps toward a comprehensive annotation system, we aimed at designing and implementing the first modules to extract meaningful features, *discover* the semantic concepts and *learn* to annotate from a set of examples.

Before starting the specific contributions of our work and presenting the results we obtained, it is worth to make a brief overview of the research background in CBIR, focusing on the main open issues.
CBIR in general refers to a broad spectrum of computer science technologies that help us to create, organize, store and efficiently access large collections of digital pictures by exploiting automatically their visual content. From this generally accepted definition it follows clearly that the scope of the research in CBIR is extremely wide. For example, it ranges from the problem of representing the semantic content of an image to the definition of suitable image similarity functions or to the automatic selection of most relevant answers to queries expressed in terms of generic visual concepts. Also, in order to cope with the many aspects of the problem an high level of expertise is required in different fields such as – for example – computer vision, machine learning and statistics, database engineering, or psychology. Indeed, as the type of queries allowed to the user by CBIR systems is becoming more and more complex – from the initial *query-by-visual-example* to the more sophisticated *query-by-keyword* and toward the long desired *fully-content-based* queries – the field requires an increasingly multidisciplinary approach to the problems; this trend was already pointed out in [RV08]. We refer the readers to [SWS+00] for a comprehensive survey of the most influential works in the area up to 2000; while [DJL+08] reports the latest developments and contains insightful comments and discussions on open issues. In [WBB+06], the authors offer an interesting panel discussion on some of the most controversial problems correlated to CBIR.

As a consequence of the above considerations, it should be clear why design and development of an effective CBIR system are considered unanimously one of the most challenging problems addressed by researchers in Computer Vision and Pattern Recognition in the last decade. Furthermore, it should come as no surprise that, despite the considerable amount of research efforts reported in the two surveys some of the crucial issues – which in our opinion are extremely intriguing research problems – still remain undressed. As anticipated above, the work described in this thesis mainly focused on one of such issues, which is the role of supervised and unsupervised learning. More specifically, we investigated the use of *unsupervised learning* techniques for the automatic definition of *semantic visual concepts*, based on which to create an algorithmic architecture comprising a pool of *supervised classifiers* for automatic annotation and, possibly retrieval.

The deployment of machine learning and statistical methods for many aspects of CBIR has emerged as an important trends in the recent years (see for example [CCMV07] and references therein). The most adopted learning paradigm is the supervised one, which proved to be effective for separating images belonging to visually well separated conceptual categories such as indoor from outdoor scenes, or cities and buildings from landscapes. The usual experimental setup is based on selected training images either containing or not the concept of interest, from which a pool of *one-versus-all* classifiers or a single *multiclass* classifier are trained. Automatic learning has been used also to build *adaptive feature vectors* (often called "signatures" in this context) from images.

Design and implementation of data dependent similarity functions have been conveniently addressed in a semi-supervised learning framework by means of feedback loops in which the user tunes the relevance of query results.

The results obtained so far in many cases by such (semi-)supervised learning modules have been satisfactory however it is widely acknowledged that the generalization capabilities reached by current CBIR systems are limited. In fact, in general system developers exploit manual annotation for indexing the images and then retrieving the proper image from the collection. Therefore, the training stage depends strongly on these manual indexes, which are used as input labels. However, although manual annotation of image content is certainly the "best case" in terms of accuracy, it is prone to (at least) two limiting factors. On one hand, it is highly subjective and the choice of the proper set of tags for an image depends not only on the objects/concepts present in the foreground of the image but also on the background context. This may result in two images containing the same object being annotated in two different ways. On the other hand, the process of annotating a large collection of image data is a time-consuming process and human users tend to make errors while performing long and repetitive tasks. Such unavoidable errors result in poorer performances of the classifiers. Indeed, it follows that the strategy adopted to create annotations may be the main drawback of many current systems.

A further issue related to the annotation process is that it worsens rapidly as the search domain goes from narrow to broad. Indeed, [SWS+00] already called attention to the crucial issue of the *scope* of the image search domain. CBIR systems designed for very narrow domains deals with images that have limited variability and better-defined visual characteristics – which are easily captured and summarized by human users in the annotation process. At the opposite extreme, quite broad domains are likely to contain highly variable subtypes of images and the related semantic concepts are more subjective and unpredictable – as consequence of which generalization is a much more challenging goal.

According to the previous analysis, we strongly believe there is a need for automatic strategies in the annotation process as viable alternatives to the traditional human-based manual annotation. Such automatization is likely to provide more flexibility to the consequent CBIR systems.

However, as opposed to manual annotation, automatic annotation in large collections of data must deal with difficult issues. First, a "broad domain" of images has a virtually unlimited and unpredictable variability in appearance even for the same semantic meaning. Another crucial point is that the user interprets an image identifying its semantic meaning by using a large amount of background and context knowledge. An automatic annotation system instead is only able to quantify and provide measurements by data processing and lacks the ability to infer information from the context. This important issue is closely related to the so-called *semantic gap* – which is almost always present in machine vision systems – consisting in a lack of coincidence between information extracted from visual data and the interpretation that the

same data has for a user in a given situation. A verbal description of a scene is contextual and depends on the knowledge of the observer whereas the image lives by itself. All these aspects are crucial and must be kept in mind during the evaluation of the performances of an annotation system.

1.2 Objectives and contributions

The long term objective of the thesis is to design an automatic system for semantic annotation and retrieval of natural images in large datasets. Specific attention is devoted to both unsupervised and supervised learning aspects, in relation to which we present two original contributions in context of *spectral methods* in statistical learning theory. Detailed descriptions of such contributions are in Chapters 4 and 5.

By integrating a suitable region-based image representation with the above learning modules, we propose an algorithmic framework for annotation and retrieval consisting of several classifiers in which classes are defined as groups of image regions labeled with common semantic keywords.
In order to create and train the system, an unsupervised learning strategy is adopted to organize the data in homogeneous clusters, and assign automatically a *tag* to all the image regions belonging to *meaningful* clusters only. Therefore, we tried to establish a direct connection between possible natural language queries and the visual appearance of spatially localized parts of images, by means of which we trained an architecture of classifiers able to automatically assign a set of labels to a given image and – consequently – to retrieve representative images belonging to the same semantic class.

In order to cope efficiently with the difficulties arising while building the above system, we designed a modular system, which is also more likely adaptable to different annotation and retrieval scenario. The main modules of the algorithmic architecture – and the specific problems connected to them – are briefly summarized in the following list.

- *Image Segmentation.* In many annotation and retrieval systems segmentation is used as preprocessing stage before extracting features. Although segmentation is widely used, highly accurate image segmentation is hard to achieve, if not impossible. There are two aspects to be considered about image segmentation. The first is that there are many possible partitions of the image and there are several correct segmentation, depending on the application. The second aspect is that in broad domains clutters and occlusions are to be expected. Segmentation algorithms providing fine decomposition could excessively fragment the scene. A weak segmentation, yielding homogeneous regions not necessarily covering entirely objects in the scene, is more adequate when handling large datasets.
  In the proposed system we obtain a weak image partition by means of a texture-color segmentation algorithm [HGS05]. This method, well-defined from a physical point of

view, measures color/texture by embedding the responses of a filter bank into a color opponent representation and exploits these measures to segment images.

- *Feature Extraction and Data Representation.* Feature extraction methods are useful to capture visual properties of an image like color, texture, position, shape and salient points.
  We create a feature vector for each segment that includes the color, texture description but also models position in the image. We can identify two advantages concerning a region-based approach. On the one hand global characterization alone cannot ensure satisfactory retrieval results since that global features are often rigid to represent an image. On the other hand local feature extraction is computationally onerous and is not well-suited to a retrieval system designed to serve a broad domain.

- *Supervised and Unsupervised Approaches to Learning.* Learning-based methods are fundamental to perform automatic annotation and to yield perceptually meaningful ranking of images as retrieval results. Clustering allows to automatically assign a set of tags to images in the absence of labels. This approach is well-suited when handling large, unstructured image repositories. In order to discover meaningful subgroups that can be likely associated to semantic concepts or sub-concepts, we determine clusters from segments previously computed.
  Therefore, for each set of clusters automatically annotated, we use a number of supervised learning machines that learns from the data to categorize all blobs extracted from the images in the database for the subsequent retrieval.

The contributions of my research work can be summarized as follows:

- to design and develop an architecture of classifiers, each representing a visual concept which could be possibly present in the image. The input feature vectors to the system are not relative to the whole image but instead they are extracted from a pool of *image parts* (which we will refer to as *blob*) obtained by coarsely segmenting the images using color and textural information.

- to study, develop and validate experimentally an algorithmic pipeline which allows for the automatic creation of training sets of blobs for each classifier by *transferring* prior knowledge given in the form of a tagging of the images.

- to study and implement an *iterative spectral algorithm*, called ν-method, which is used as the algorithmic characterization of the visual concepts. ν-method belongs to a class of spectral classification algorithms which have been shown in [LGRO$^+$08] to obtain good results when compared with a number of more popular state-of-the-art classifiers.

- to use the above methods to make the first step towards a complete CBIR engine. Preliminary tests are performed on the standard benchmark database called COREL30K.

The results of the experiments are promising, albeit preliminary thus requiring more extensive analysis.

1.3 Organization of the thesis

The structure of the thesis is organized as follows:

- **Chapter 2** is devoted to introduce visual cues and segmentation approaches in the image domain. We discuss state-of-the-art of image representation methods focusing on those adopted in our work, mostly based on color and texture information. We also present a taxonomy of some popular methods for image segmentation. Among these, we give particular emphasis and details on feature-based segmentation techniques.
- **Chapter 3** briefly presents the relevant background pertaining to supervised in opposition to unsupervised learning techniques. We provide the main ingredients of supervised learning and introduce the regularization theory. Among unsupervised learning methods, the focus is on two popular data clustering techniques that we adopted in the pipeline of the system.
- **Chapter 4** gives an overview of the whole procedure to infer semantic concepts to training images. Starting from a general discussion on the setting, we move to explain the requirements arising from the algorithmic standpoint. We introduce the stages to obtain image segmentation, feature vectors, and homogeneous clusters of blobs, then the approach to label each cluster. We also show results of this totally unsupervised and automatic procedure for a well-known large dataset.
- **Chapter 5** shows how a large class of regularization methods, collectively known as spectral regularization and originally designed for solving ill-posed inverse problems, gives rise to regularized learning algorithms. We present several examples of spectral algorithms for supervised learning and discuss similarities and differences between the various methods. Finally we comment on their properties in terms of algorithmic complexity.
- **Chapter 6** is devoted to experimentally validate the algorithms discussed in Chapter 5, showing the effectiveness of the proposed approach. We apply them to a number of classification problems on sets of well known benchmark data, comparing the results with the ones reported in the literature; then we consider a more specific application, face detection, analyzing the results provided by a spectral regularization algorithm as opposed to a well-known learning technique. The second part of the chapter is devoted to present a number of preliminary experiments using a large dataset to test the potential of the spectral algorithms in the context of automatic image annotation and retrieval.

Chapter 2

Feature-based representation and segmentation

Most CBIR systems perform feature extraction as a preprocessing step. Once obtained, visual features act as inputs to subsequent image analysis tasks, such as similarity estimation, concept detection, or annotation. As previously discussed, the goal here is to automatically describe the content of images. To do so, the image has to be represented in a suitable way possibly discarding unnecessary information and the following step is to statistically assign it to a category. This can be done only if an appropriate similarity measure is defined.

The simplest way to represent image information is to consider the image at pixel-level representation, but this choice is often not optimal because it does not emphasize any peculiarity of the considered image. Alternatively, one can take into account a list of features based either on global or local properties. Typically, we expect that a good representation for a certain task is the best compromise between loss of information and enhancement of image characteristics.

In this chapter we briefly present visual low-level features and their application to image segmentation. An introduction to basic concepts and methodologies on image features and segmentation approaches can be found in [GW08]; surveys on image feature extraction in CBIR system can be found in [RHC99] and [DJL+08]. In Section 2.1, we briefly review the state-of-the-art of image representation methods focusing on those adopted in our work, mostly based on color and texture information. In Section 2.2 we give an overview of some approaches to compute similarity between images when one of the description discussed in the previous sections is adopted.

In Section 2.3 we present a taxonomy for image segmentation focusing on feature-based methods.

2.1 Feature-based representation

A good feature should capture a certain visual property of an image, either globally for the entire image or locally for a small group of pixels.

The feature is defined as a function of one or more measurements, each of which specifies some quantifiable property of an object. We classify the various features currently employed as follows:

- *Local features*: features calculated over the results of subdivision of the image band on image segmentation or edge detection. In a local description a pixel is represented by a set of features extracted in its neighborhood (e.g., average color values across a small block centered around the pixel).

- *Global features*: features calculated over the entire image or just regular sub-areas of an image. For instance, in a color layout approach, an image is divided into a small number of sub-images and the average color components (e.g., red, green, and blue intensities) are computed for every sub-image. The overall image is thus represented by a vector of color components where a particular dimension of the vector corresponds to a certain sub-image location.

The advantage of global extraction is its high speed for both extracting features and computing similarity. However, global features are often too rigid to represent an image. Specifically, they can be oversensitive to location and hence fail to identify important visual characteristics. To increase the robustness to spatial transformation, an alternative approach relies on local extraction followed by a further step of feature summarization.

An alternative categorization of image features can be based on the source of information extracted. *Low-level* features are extracted directly from digital representations of objects, have little or nothing to do with human perception and can be extracted from the original images. *High-level* features are computed from basic data or low-level features; they represent a higher level of abstraction and they are typically more concerned with the system as a whole and its goals.

In many applications, the representation of image data does not rely only on one type of cue. Conversely, two or more different cues are extracted, resulting in two or more corresponding vectors of features, one for each given image. A common practice is to organize the information provided by all these cues as the elements of one single vector, commonly referred to as a *feature vector*. The set of all possible feature vectors constitutes a feature space.

In this section we report some of the most common methods for describing the image content via set of features. An exhaustive overview is out of the scope of this thesis, here we mainly focus on the most popular cues and descriptors that have been applied in the automatic

annotation and retrieval context (see, for example, [Man96], [MrOVY01] and references in [DJL+08]): color, texture and shape.

2.1.1 Color representation

Color is perhaps the most expressive of all the visual cues and has been extensively studied in the image retrieval research.

The visual experience of the normal human eye is not limited to gray scale, therefore *color* is an extremely important aspect of digital imaging. In a very general sense, color conveys a variety of rich information that describes the quality of objects. The perception of color is allowed by the color-sensitive neurons known as *cones* that are located in the retina of the eye. The cones are responsive to normal light and are distributed with greatest density near the center of the retina, known as *fovea*. The *rods* are neurons that are sensitive at low-light levels and are not capable to distinguish color wavelengths. They are distributed with greatest density around the periphery of the fovea, with very low density near the line of sight. In the normal human eye, colors are sensed as near-linear combinations of long, medium and short wavelengths which roughly correspond to the three *primary colors* that are used in the standard video camera systems: Red (R), Green (G) and Blue (B).

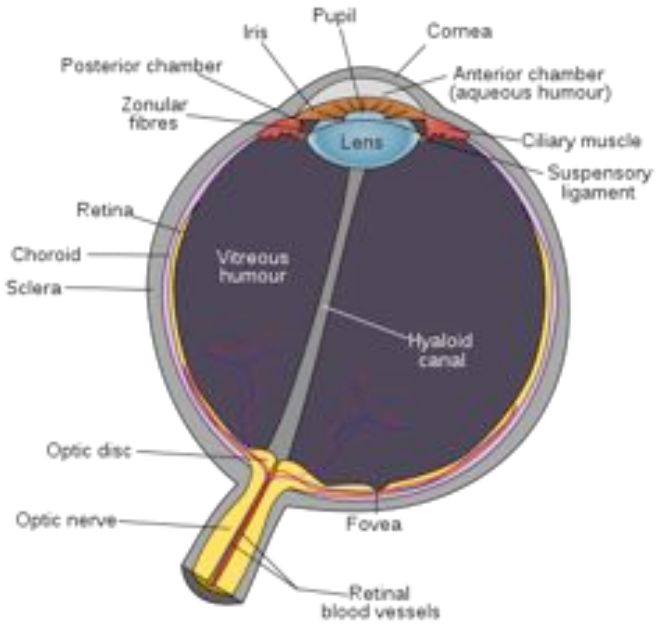


Figure 2.1: Schematic diagram of the human eye.

As a consequence of the affinity between the human visual system, the RGB model seems a natural way to represent color. Conversely, the RGB model is not the optimal choice in several applications, as commented in the next section. The state-of-the-art on color includes a number of different spaces and metrics and it would be almost impossible to mention all of them here. In the next two sections we first focus on *non-linear* color models, because they lend to mimic the higher level processes which allow the perception of color of the human visual system and then we will discuss in details how to represent the color information.

Color spaces

Many *color spaces* have been defined over the years: CIE 1931, CIE $L^*u^*\ v^*$, CIE $L^*a^*b^*$, RGB, sRGB, Adobe RGB, YIQ/YUV/YCbCr, HSV, HSL, CMYK, hue-min-max-difference (HMMD) [MrOVY01], just to name the most used. Each of them gives rise to different image representations that enhance or attenuate certain color features.

The last decade witnessed an increasing interest on exploiting the color spaces that seem to better coincide with human vision than basic RGB color space. Among them, non-linear color representations take into account the non-linearity in the response of the human eye to light. There are two groups of non-linear color spaces widely used that "imitate" human vision. The HSV/HSI/HSB/HSL color spaces separate the color into *Hue*, describing the dominant wavelength, *Saturation* referring to the dominance of hue and *Value*, the intensity. The other group is the CIE $L^*a^*b^*$ and CIE $L^*u^*\ v^*$ color spaces that separate color into luminance and two color coordinates in an effort to create a color space which is *perceptually uniform*.

In this thesis we will focus on two color spaces belonging to these groups: HSV and CIE $L^*a^*b^*$ spaces.

HSV model. The RGB space does not attempt to describe color relationships as human visual system; in this sense color is better represented in terms of hue, saturation and intensity.

The representation of the HSV space is derived from the RGB space cube, with the main diagonal of the RGB model, as the vertical axis in HSV. As saturation varies form 0.0 to 1.0, the colors vary from unsaturated to saturated white component. Hue ranges from 0 to 360 degrees, with variation beginning with red, going through yellow, green, cyan, blue and magenta and back to red.

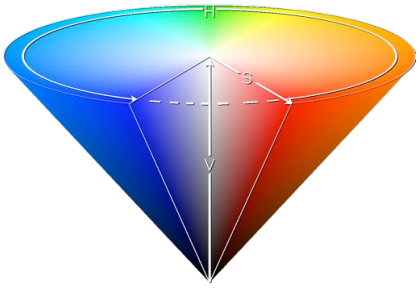


Figure 2.2: The conical representation of the HSV space

HSV is intuitively corresponding to the RGB model from which they can be derived through the following non-linear transformations:

$$H = \begin{cases} \theta & B \leq G \\ 360 - \theta & B > G \end{cases}$$

with

$$\theta = \cos^{-1}\left\{ \frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\sqrt{(R-G)^2+(R-B)(G-B)}} \right\},$$

$$S = 1 - \frac{3\left[\min(R,G,B)\right]}{R+G+B},$$

$$V = \frac{1}{3}(R+G+B).$$

The graphical representation of the Hue-Saturation-Value (HSV) model (see Figure 2.2) consists in a circular cone to describe the space. Pure black lies at the tip of the cone, pure white at the center of the base, and pure hues around the perimeter of the base.

However, HSV space is not perceptually uniform, i.e. differences among colors perceived by the human eye as being of the same entity are not mirrored by similar distances between the points representing those color in such spaces.

CIE $L^*a^*b^*$ model. The *uniform color spaces* overcome the problem afore-mentioned: the most widely used are the CIE (*Commission International de l'Eclairage*) spaces. CIE $L^*a^*b^*$ and CIE $L^*u^*v^*$ spaces are two typical color systems developed to represent perceptual uniformity and thus meets the psychophysical need for the human observer. They are opponent

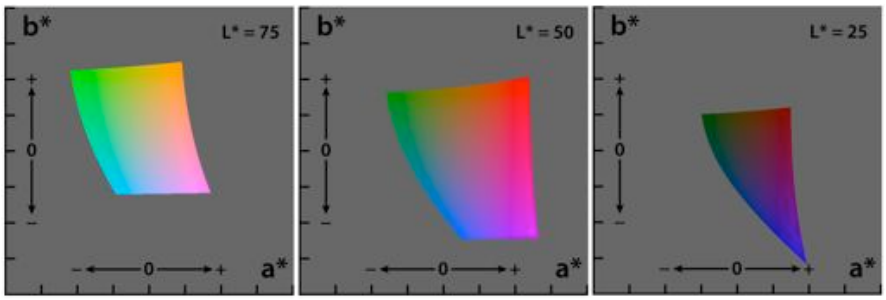


Figure 2.3: The CIE $L^*a^*b^*$ color space, showing only colors that fit within the sRGB gamut (and can therefore be displayed on a typical computer display).

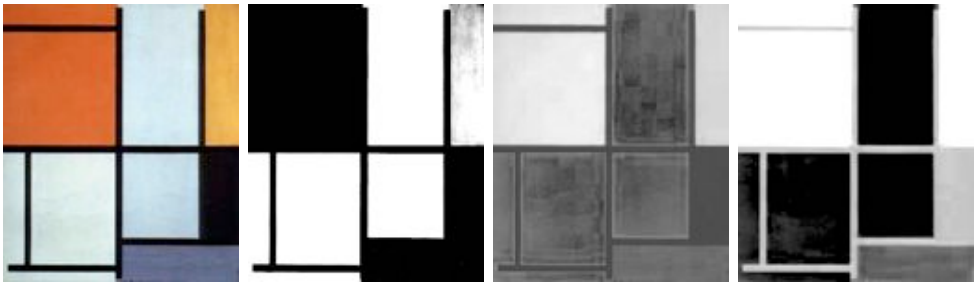
color spaces in which Euclidean distances between colors approximately reflect perceived color differences.

The CIE $L^*a^*b^*$ color space is a *color-opponent* space with L for lightness and a and b for the color-opponent dimensions, based on nonlinearly-compressed CIE XYZ color space coordinates[1]. It is derived from the "master" space CIE XYZ color space, which can predict which spectral power distributions will be perceived as the same color, but which is not particularly perceptually uniform. The intention of CIE $L^*a^*b^*$ is to create a space which can be computed via simple formulas from the XYZ space, but is more perceptually uniform than XYZ. The three coordinates of CIE $L^*a^*b^*$ represent the lightness of the color (L* = 0 yields black and L* = 100 indicates diffuse white; specular white may be higher), its position between red and green (a*, negative values indicate green while positive values indicate red) and its position between yellow and blue (b*, negative values indicate blue and positive values indicate yellow). The asterisks (*) after L, a and b are part of the full name, since they represent L*, a* and b*, to distinguish them from Hunter's L, a and b[2].

Since the CIE $L^*a^*b^*$ model is a three-dimensional model, it can only be represented properly in a three-dimensional space. Two-dimensional depictions are chromaticity diagrams: sections of the color solid with a fixed lightness. The visual representations of the full gamut of colors in this model is never accurate; they are here just to help in understanding the concept (see Figure 2.3).

[1] CIE XYZ is an absolute color space (not device dependent). The X,Y,Z values are not the physically observed red, green, blue colors. The X,Y,Z values are a sort of "derived" parameters from the red, green, blue colors.

[2] Both the Hunter Lab scale and the CIE $L^*a^*b^*$ scale are visually meaningful; the CIE $L^*a^*b^*$ is an improvement to the final version of Hunter Lab (1966). The formulas for Hunter Lab are square roots using CIE XYZ, whereas CIE $L^*a^*b^*$ is calculated using cube roots of XYZ.



(a) input (b) L* (c) a* (d) b*

Figure 2.4: An example of color image with corresponding CIE $L^*a^*b^*$ channels.

Color descriptors

Research on color features has focused also on the summarization of color values in a set of descriptors. Color features or descriptors include color moments and plain histograms [SB91] but also several features to discriminate among images with similar histograms but very different visual appearance.

Histograms. It is often advantageous to "compress" or approximate an original distribution by another distribution with a more compact description. This yields important savings in storage and processing time and a certain perceptual robustness to the matching. Distributions are usually compressed by partitioning the underlying space into a fixed number of bins, usually of a predefined size: the resulting quantized data structure is a *histogram*.

In content-based retrieval, histograms have mostly been used in conjunction with color features, but there is nothing against being used in texture or local geometric properties.

Color histogram describes the distribution of colors in the whole image or within a region of interest. The histogram is invariant to rotation, translation and scaling of an object but does not contain semantic information, therefore two images with similar color histograms can possess different contents. Formally a histogram H for a given image is defined as a vector $H = \{h_1, h_2, \ldots h_i, \ldots, h_N\}$ where i represents a color interval, h_i is the number of image pixels of color i, and N is the number of bins, i.e. the number of colors in the adopted color model. In order to compare images of different size, color histograms should be normalized. Figure 2.5 shows an example of a two dimensional normalized color histogram.

The extensive application of color histograms to retrieval problems is due their ease and efficiency in computation; moreover, they achieve significant data reduction, and they are robust to noise and local image transformation. However, as a feature vector for image retrieval, it is susceptible to false positives, since it does not capture spatial image information. This problem is especially acute for large databases where false positive are more likely to

occur. Several CBIR systems introduce spatial information to improve retrieval performance exploiting extension of basic histograms.

Histogram extensions. Several schemes for jointly using spatial and color information in histogram based descriptions have been proposed. One common approach is to divide images into sub-regions and impose positional constraints on the image comparison (*image partitioning*). Another approach is to augment histograms with local spatial properties (*histogram refinement*). Smith and Chang [SC96] partition an image into binary color sets. They first select all colors that are "sufficiently" present in a region. The colors for a region are repre-

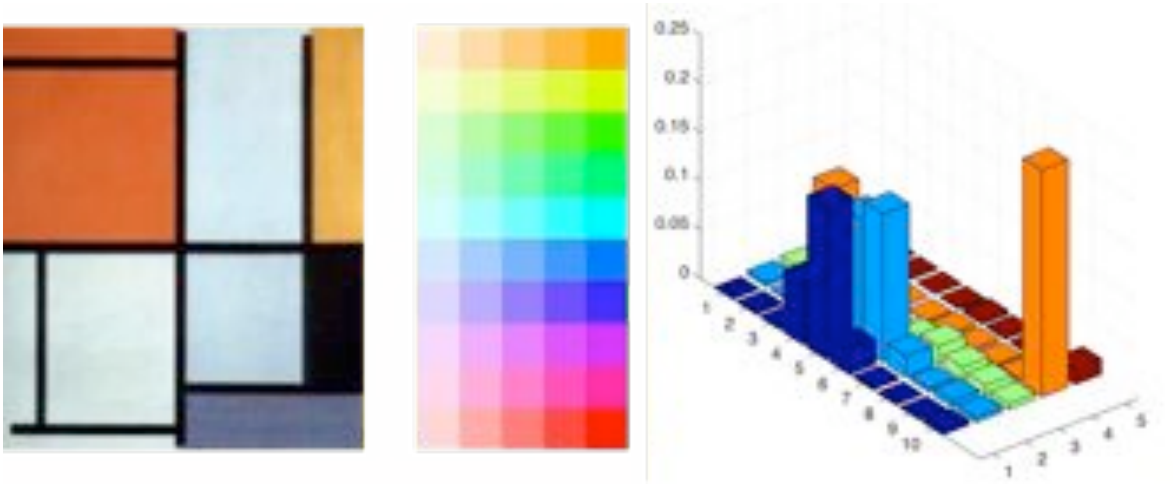


Figure 2.5: A quantization of the HS space (H = 10, S = 5) and a HS histogram example. For the sake of simplicity we depict the HS quantization with a rectangle, even if it should be a circle (H indicates the angle).

sented by a binary color set that is computed using histogram backprojection [SB91]. The binary color sets and their location information constitute the feature. Stricker and Dimai [SD96] divide an image into five fixed overlapping regions and extract the first three color moments of each region to form a global feature vector. The storage requirements for this method are low. The use of overlapping regions makes the feature vectors relatively insensitive to small rotations or translations. Pass and Zabih [PZ96] use another approach. They partition histogram bins by the spatial coherence of pixels. A pixel is coherent if it is a part of some "sizable" similar-colored region, and incoherent otherwise. A *color coherence vector* (CCV) represents this classification for each color in the image. CCVs are fast to compute and appear to perform better than histograms. Huang et al. [HKM+97] define color feature for image indexing/retrieval called the *color correlogram*. Color correlograms characterize color distributions of pixels and spatial correlation of pairs of colors. A set of color descriptors tested for inclusion in the MPEG-7 standard, and well suited to natural images and

video, is described in [MrOVY01]. These include histogram-based descriptors, spatial color descriptors, and texture descriptors suited for retrieval.

Another extension of histograms is the *multiresolution histogram* which is mainly used to describe shape and texture [HGN04]. In fact, as mentioned above, histograms have mostly been used in conjunction with color features, but they are frequently used to summarize texture and geometric properties especially when histogram encodes some further information. The disadvantages of treating histograms simply as vectors of frequencies are noted in [RTG00]. The main issue is that the vector representation ignores the location of bins used to generate the histogram. In the next paragraph we will describe *signature*, a feature descriptor recently used in retrieval context [RTG98], that overcomes this problem.

Signatures. A drawback of histogram representation is that, even when the binning is adaptive, often only a small fraction of the bins in a histogram contain significant information. For instance, when considering color, a picture of a desert landscape contains mostly blue pixels in the sky region and yellow/brown pixels in the rest. A finely quantized histogram in this case is highly inefficient. On the other hand, a multitude of colors is a characterizing feature for a picture of a carnival in Rio, and a coarsely quantized histogram would be inadequate. In brief, because histograms are fixed-size structures, they cannot achieve a balance between expressiveness and efficiency. In contrast, signatures identify dominant clusters that are extracted from the original distribution using a clustering algorithm such as vector quantization, and are used to form its compressed representation.

A signature is a set of the main clusters or modes of a distribution, each represented by a single point (the cluster center) in the underlying space, together with a weight that denotes the size of that cluster. Simple images have short signatures, complex images have long ones. Of course, in some applications, fixed-size histograms may still be adequate, and can be considered as special cases of signatures. A *signature* $\{\mathbf{s}_j = (\mathbf{m}_j, w_j)\}$, on the other hand, represents a set of feature clusters. Each cluster is represented by its mean (or mode) m_j, and by the fraction w_j of pixels that belong to that cluster. The integer subscript j ranges from one to a value that varies with the complexity of the particular image. While j is simply an integer, the representative $\mathbf{m}_j$ is a d-dimensional vector. Therefore, a histogram H can be viewed as a signature $\{\mathbf{s}_j = (\mathbf{m}_j, w_j)\}$ in which the vectors i index a set of clusters defined by a fixed a priori partitioning of the underlying space. If vector i maps to cluster j, the point $\mathbf{m}_j$ is the central value in bin i of the histogram, and w_j is equal to h_i.

2.1.2 Texture features and descriptors

Texture features are designed to capture the granularity and the repetitive patterns of surfaces within a picture. Texture is a property of areas, hence, it is contextual property and its definition must involve pixel values in a spatial neighborhood. Tuceryan and Jain [TJ98]

identify four major categories of features for texture identification: *statistical*, *geometrical*, *model-based* and *signal processing* features. In image processing the statistical model-based techniques are commonly used. The use of statistical features is one of the early methods proposed in the machine vision literature and a large number of texture features have been proposed. Due to the extensive research on texture analysis over the past 30 years it is impossible to list all published methods, but the following description provides short introduction to each of the above categories, together with some key references.

- Statistical methods

  Statistical methods analyze the spatial distribution of gray values, by computing local features at each point in the image, and deriving a set of statistics from the distributions of the local features. Depending on the number of pixels defining the local feature statistical methods can be further classified into first-order (one pixel), second-order (two pixels) and higher-order (three or more pixels) statistics. The basic difference is that first-order statistics estimate properties (e.g. average and variance) of individual pixel values, ignoring the spatial interaction between image pixels, whereas second and higher-order statistics estimate properties of two or more pixel values occurring at specific locations relative to each other. The most widely used statistical methods are co-occurrence features [HSD73] and gray level differences [WDR76], which have inspired a variety of modifications later on.

- Geometrical methods

  Geometrical methods consider texture to be composed of texture primitives, attempting to describe the primitives and the rules governing their spatial organization.

  The primitives may be extracted by edge detection with a Laplacian-of-Gaussian or Difference-of-Gaussian filter, [TJ90], [VP87], [TJ90] by adaptive region extraction [TT90], or by mathematical morphology [Ser82].

  Once the primitives have been identified, the analysis is completed by computing statistics of the primitives (e.g. intensity, area, elongation, and orientation).

  The structure and organization of the primitives can also be presented using Voronoi tessellations [TJ90]. Image edges are an often used primitive element. In [DCA79] are defined *generalized co-occurrence matrices*, which describe second-order statistics of edges.

- Model-based methods

  Model-based methods hypothesize the underlying texture process, constructing a parametric generative model, which could have created the observed intensity distribution. The intensity function is considered to be a combination of a function representing the known structural information on the image surface and an additive random noise sequence [MC93].

- Signal Processing methods

  Signal processing methods analyze the frequency content of the image. Spatial domain filters such as local linear transforms and various masks designed for edge detection are the most direct approach for capturing frequency information. Another class of spatial filters are moments which correspond to filtering the image with a set of spatial masks. The resulting images are then used as texture features.

  "True" frequency analysis is done in the Fourier domain. The Fourier transform describes the global frequency content of an image, without any reference to localization in the spatial domain, which results in poor performance. Spatial dependency is incorporated into the presentation with a window function, resulting in a so-called short-time Fourier transform. The squared magnitude of the two-dimensional version of the short-time Fourier transform is called *spectrogram*, which is used in [BL76] in analyzing shape from texture.

  Multiresolution analysis, the so-called *wavelet transform*, is achieved by using a window function, whose width changes as the frequency changes [Mal89]. If the window function is Gaussian, the obtained transform is called the Gabor transform A two-dimensional Gabor filter is sensitive to a particular frequency and orientation. Other spatial/spatial-frequency methods include the Difference-of-Gaussians and the pseudo-Wigner distribution.

  Texture description with these methods is done by filtering the image with a *bank of filters*, each filter having a specific frequency (and orientation). Texture features are then extracted from the filtered images.

  Often many scales and orientations are needed, which results in texture features with very large dimensions. Dimensionality can be reduced by considering only those bands, which have high energy [JF91]. Alternatively, redundancy can be reduced by optimizing filter design so that the frequency space is covered in a desired manner. For a review and a comparative study of the most major filtering approaches see [RH99].

Wavelet models (Haar wavelets)

Wavelets (see [Dau92], [Add02], [SDS95] and [Mal89]) are a mathematical tool for hierarchically decomposing functions. They allow a function to be described in terms of a coarse overall shape, plus details that range from broad to narrow. Regardless of whether the function of interest is an image, a curve, or a surface, wavelets offer an elegant technique for representing the levels of detail present. In the signal processing field, most wavelet-based approaches make reference to *discrete wavelet transform* (DWT), that is a discretely sampled version of wavelet transform. Among all the possible basis (see for instance [Dau88]), Haar wavelets [A.10] are maybe the most commonly used DWT in image processing thanks to their ability to describe 2D signals in a simple way.

To get a sense for how Haar wavelets work, let us start with an example. Suppose we are given a one-dimensional "image" with a resolution of four pixels, having values

$$[\ 9 \quad 7 \quad 3 \quad 5\ ]$$

We can represent this image in the Haar basis by averaging the pixels together, pairwise, to get the new lower resolution image with pixel values

$$[\ 8 \quad 4\ ]$$

Clearly, some information has been lost in this averaging process; to recover the original four pixel values from the two averaged values, we need to store some *detail coefficients*, which capture the deviations of pixel values from the mean

$$[\ 1 \quad -1\ ]$$

We can then recursively repeat the averaging, to obtain the final decomposition

$$[\ 6\ ]$$

The Haar wavelet transform of the original four-pixel image is the single coefficient representing the overall average of the original image, followed by the detail coefficients in order of increasing resolution. Thus, for the one-dimensional Haar basis, the wavelet transform of our original four-pixel image is given by

$$[\ 6 \quad 2 \quad 1 \quad -1\ ]$$

In this process there is no gain or loss of information, the original image had four values, and so does the transform. It is possible to reconstruct the image to any resolution by recursively adding and subtracting the detail coefficients from the lower resolution versions. One big advantage of the wavelet transform is that often a large number of the detail coefficients turn out to be very small in magnitude: removing these small coefficients from the representation introduces only small errors in the reconstructed image, giving a form of "lossy" image compression.

So far we presented the Haar wavelet transform for a 1D image; the transform can be easily extended to 2D images by performing two subsequent steps: we first apply the one-dimensional wavelet transform to each row of pixel values; this operation gives us an average value along with detail coefficients for each row. Next, we treat these transformed rows as if they were themselves an image and apply the one-dimensional transform to each column. The resulting

Figure 2.6: 2D wavelet transform (image from [SDS95]).

values are all detail coefficients except for a single overall average coefficient. Figure 2.6 illustrates the two steps of the transform.

Examples of texture feature using wavelet transform for retrieval purposes can be found in [DV02] and [KBC04].

Gabor filters

Gabor filters [Gab46] have received a lot of attention by the computer vision community since they have been shown to model the spatial summation properties of the receptive fields of the so called "bar cells" in the primary visual cortex. They have been successfully applied to a number of problems involving the modeling of texture properties of image regions [DH95], [JF91], [JRL97], [GD03].

A Gabor function can be viewed as a sinusoidal plane modulated by a Gaussian envelope. Figure 2.7 shows the components of a 1D function.

Definition (2D Gabor filter) A 2-Dimensional Gabor filter can be formulated as:

$$G(x,y) = e^{-\frac{\hat{x}^2 + \gamma^2\hat{y}^2}{2\sigma^2}} cos\left(2\pi\frac{\hat{x}}{\lambda}\right)$$

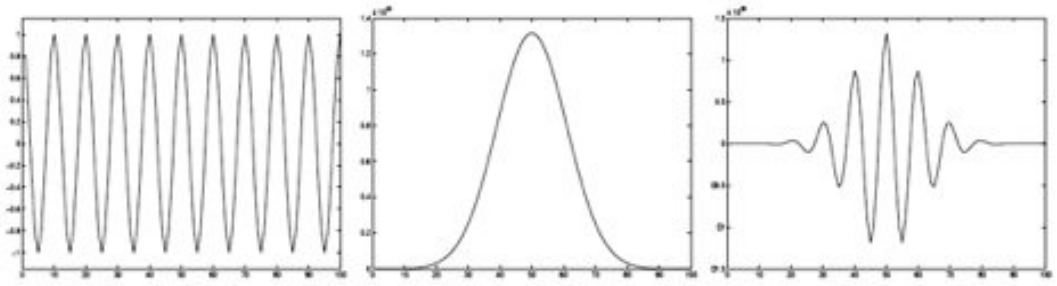


Figure 2.7: Components of a 1D Gabor filter: sinusoidal, Gaussian and the resulting Gabor filter.

where

$$\hat{x} = xcos\theta + ysin\theta$$

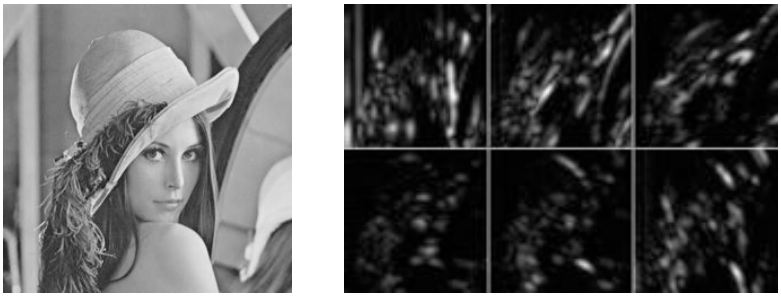
$$\hat{y} = -xsin\theta + ycos\theta$$

The parameters involved in the construction of a 2D Gabor filter are:

- the variance of the Gaussian function σ,
- the wavelength of the sinusoidal function λ,
- the orientation of the normal to the parallel stripes of the Gabor function θ and
- the spatial aspect ratio γ, which specifies the ellipticity of the support of the Gabor function. For $\gamma = 1$, the support is circular. For $\gamma < 1$ the support is elongated in the orientation of the parallel stripes of the function.

Using different values for wavelength, variance of Gaussian and orientation, one can design a filter that is responsive to gradients of a given thickness and orientation. Usually in image processing a set of Gabor filters is designed to capture different types of information, for example, Figure 2.8.b shows the responses of a set of Gabor filters with six different orientations, computed on the image 2.8.

Since Gabor wavelet proves to be very useful in texture analysis, a number of approaches for browsing and retrieval of image data were proposed using Gabor filters (see, for example, [Man96], [ZWIL00] and [CD05]).



a *b*

Figure 2.8: (a) The original image. (b) The responses of Gabor filters at six different orientations.

2.1.3 Shape descriptors

The shape of an object is an important and basic visual feature for describing image content [ZL04]. Shape representation generally aims at effective and perceptually important shape features based on boundary information – *contour-based methods* and/or on interior content – *region-based methods*. Each class can be further broken into *structural* and *global approaches*, depending on whether the shape is represented as a whole or by segments or sections [ZL04]. Common simple global descriptors are *area*, *circularity* (perimeter^2/area), *eccentricity* (length of major axis/length of minor axis), *major axis orientation*, and *bending energy* [YWB74]. These simple global descriptors usually can only discriminate shapes with large differences, therefore, they are usually used as filters to eliminate false hits or combined with other shape descriptors to discriminate shapes. Since we do not include shape in the blob representation (see Chapter 4) we give a list and references of some shape descriptors/similarities used recently in the retrieval literature.

In general, over the years we have seen a shift from global shape representations (e.g., in [FSN$^+$95]) to more local descriptors (e.g., in [MG95], [BBP00], and [PDM02]) due to the typical modeling limitations. Representation of shape using discrete curve evolution to simplify contours is discussed in [LL00]. This contour simplification helps to remove noisy and irrelevant shape features from consideration. A new shape descriptor for similarity matching, referred to as *shape context*, is proposed, which is fairly compact yet robust to a number of geometric transformations [BMP02]. In [BBP00], curves are represented by a set of segments, or tokens, whose feature representations (curvature and orientation) are arranged into a metric tree [CPZ97] for efficient shape matching and shape-based image retrieval. A dynamic programming (DP) approach to shape matching is proposed in [PDM02], where shapes are approximated as sequences of concave and convex segments. One problem with this approach is that the computation of Fourier descriptors and moments is slow, although pre-computation may help produce real-time results. Continuing with Fourier descriptors, exploitation of both

the amplitude and phase, as well as the use of dynamic time warping (DTW) distance instead of Euclidean distance, is shown an accurate shape matching technique in [BCP05]. The rotational and starting point invariance otherwise obtained by discarding the phase information is maintained here by adding compensation terms to the original phase, thus allowing its exploitation for better discrimination.

2.2 Similarity measures

A different road to assigning a meaning to an observed feature set, is to compare a pair of observations by a similarity function.

Several measures have been proposed for the dissimilarity between two point sets (histograms, signatures, etc.) $H = \{h_i\}$ and $Q = \{q_i\}$. In the following we present some of the most popular similarity measures.

- *Minkowski-Form* distance:

$$d_{L_r}(H, Q) = \left(\sum_i |h_i - q_i|^r \right)^{\frac{1}{r}}$$

  The L_1 distance is often used for computing dissimilarity between color images ([SB91]). Other common usages are L_2 and L_∞ distances.

- Swain and Ballard [SB91] proposed the use of the *intersection* distance between histograms:

$$d_\cap(H, Q) = 1 - \frac{\sum_{i=1}^{n} \min(h_i, q_i)}{\sum q_i}$$

  The intersection is attractive because of its ability to handle partial matches when the areas of the two histograms (the sum over all the bins) are different. It is also proved in [SB91] that if all images have the same number of pixels, then this distance has the same ordinal properties as the L_1 distance.

- the *Kullback-Leibler* distance is often used to measure the disparity between distributions. However, the K-L divergence is non-symmetric and is sensitive to histogram binning. The empirically derived *Jeffrey divergence* is a modification of the K-L divergence that is numerically stable, symmetric and robust with respect to noise and the size of histogram bins. It was proposed by [PRTB99] and it is defined as follow:

$$d_j(H, Q) = \sum_i \left(h_i \log \frac{h_i}{m_i} + \log \frac{q_i}{m_i} \right)$$

  where $m_i = \frac{h_i + q_i}{2}$

- χ^2 *Statistics*:

$$d_{\chi^2}(H,Q) = \sum_i \frac{(h_i - m_i)^2}{m_i}$$

- *Quadratic-form* distance was suggested in [NBA93]. The distance between two vectors (or, more precisely, histograms) is defined as follows:

$$d_\Sigma(H,Q) = \sqrt{(\mathbf{h}-\mathbf{q})^T \Sigma (\mathbf{h}-\mathbf{q})}$$

  where $\mathbf{h}$ and $\mathbf{q}$ are vectors that list all the entries in H and Q. The similarity between two elements (*ground distance*) is incorporated, by means of a similarity matrix $\Sigma = [\sigma_{ij}]$ where σ_{ij} denotes similarity between bins i and j. Here i and j are sequential (scalar) indices into the vectors.

- The *earth mover's distance* (EMD) [RTG00] proposed another soft matching scheme for signatures in the form of sets of vectors. The measure treated the problem of image matching as one of moving components of the signatures (see Section 2.1.1) of images from one to the other, with minimum effort, synonymous with moving earth piles to fill holes.
  Given two signatures $P = \{(p_1, w_{p_1}), \ldots, (p_n, w_{p_n})\}$ and $K = \{(k_1, w_{k_1}), \ldots, (k_m, w_{k_m})\}$ the EMD is defined as follow:

$$EMD(P,K) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\sigma_{ij}f_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{n}f_{ij}}$$

  where σ_{ij} is the distance between the clusters p_i and k_i, and f_{ij} is the optimal flow between the two signatures obtained by solving the transportation problem [RTG00].

In Figure 2.9 we show a simple application of the similarity measures presented in this section.

2.3 Image segmentation

In many annotation and retrieval systems segmentation is used as preprocessing stage before extracting region-based features. A segmentation algorithm concerns with partitioning images by determining disjoint and homogeneous regions or, equivalently, finding edges or boundaries. The literature on segmentation is very rich (see for example [SK94] and references therein). In this overview we follow a taxonomy suggested in [LM01]. They divide the segmentation algorithms into:

1. **feature space-based techniques**: they are based on the idea that pixels could manifest themselves as *clusters* or peaks into *histograms*. These two approaches share a

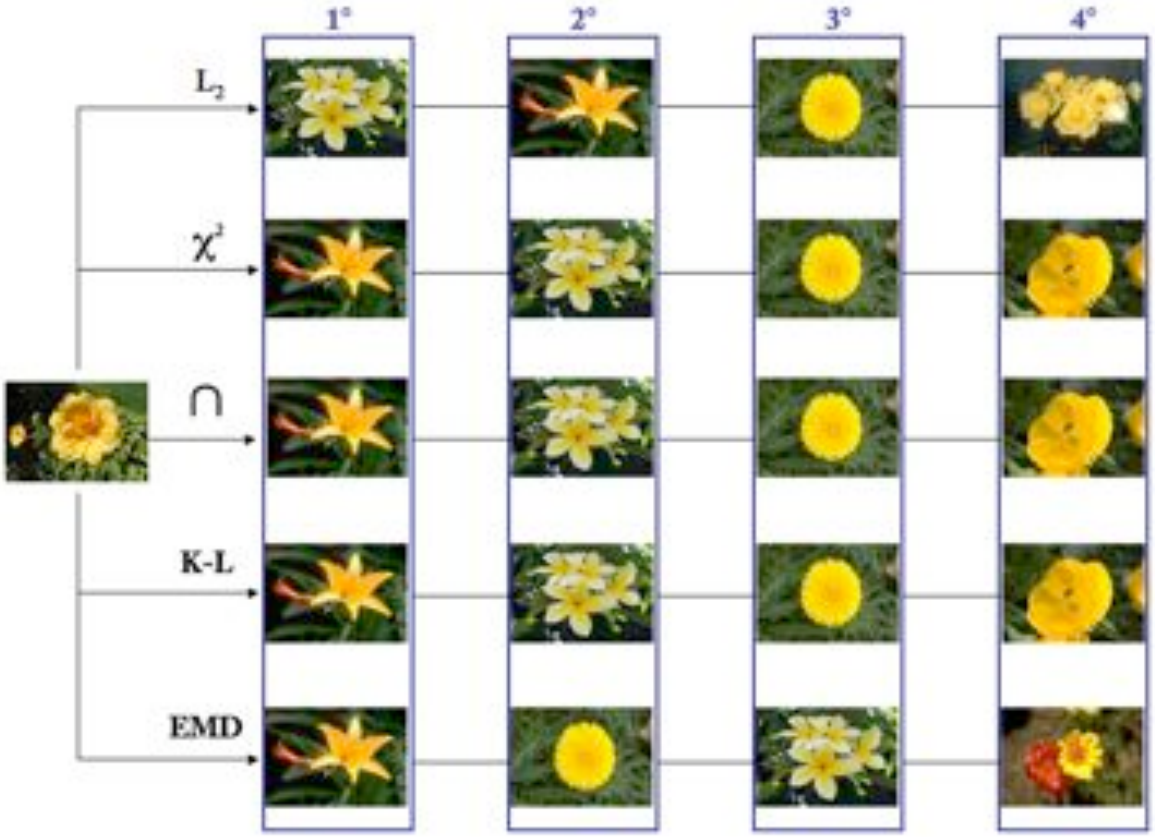


Figure 2.9: Examples of retrieval using different similarity measures between HS-histograms. The first column shows the query image. Other columns identify four nearest neighbors for each similarity measure. We applied the 2D histogram shown in Figure 2.5.

common property: they work in a certain feature space, which may be one of the color spaces or a space induced by other attributes, and they generally neglect the spatial relationships. For this reason, it is necessary to apply a connected component labeling to detect the unconnected regions at the end of the segmentation process.

2. **image-domain based techniques**: the feature space-based techniques do not guarantee that regions also show spatial compactness, which is a desirable property in segmentation applications beside homogeneity. In fact, cluster analysis and histogram thresholding do not account for the spatial locations of pixels; the description they provide is global and it does not exploit the fact that points of a same object are usually spatially close due to surface coherence. Usually, the available techniques can be divided into three main groups: *Split and Merge* [HP76], *Region Growing* [BF77] and *Edge-based techniques* [GKRR01], [KWT88] and [OS88]. The first method is iterative: it starts with

an initial inhomogeneous partition of the image (usually the initial segment is the image itself) and it keeps performing splitting and merging until homogeneous partitions with respect to a certain statistic criterion are obtained. In the second approach an homogeneous region of an image may be obtained through a growth process which, starting from a preselected seed, progressively agglomerates points around it satisfying a certain homogeneity criterion. Finally, *edge-based techniques* provide the segmentation by detecting the edges among regions. The central challenge for edge detection techniques is to find procedures that produce closed contours around the objects of interest.

3. **physics based techniques**: the algorithms examined so far are certainly prone to segmentation errors if the objects in a image are affected by highlights and shadows. These phenomena cause the appearance of grey-levels or color change, more or less drastically, on uniform surfaces. The algorithms above mentioned are very likely to return over-segmented regions. The only way to overcome this drawback is to analyze how light interacts with materials and to introduce models of this physical interaction in the segmentation algorithms (see, for example [KSK90], [BLL89]).

Feature space-based segmentation

Histogram thresholding *Histogram thresholding* is one of the most popular techniques for segmenting grey-level images. This is primarily due to the fact that it is an easy-to-implement and efficient method and provides satisfactory results in many cases. In various applications, it can be effectively used as the initial step of more sophisticated image analysis tasks. Examples of such application include segmentation of brain tissue and/or tumors in magnetic resonance (MR) images and quantification of cell nuclei and chromosomes in microscope images. The basic method is trivial: the peaks and the valleys of the 1D brightness histogram can be easily identified, respectively, with objects and backgrounds of the grey-level images. The threshold for splitting the image can be arbitrarily fixed or chosen with respect to a given, usually statistic, automatic criterion. The most famous approaches are the *peaks and valleys method* and the *global threshold selection*. The first one locates the two higher peaks and afterwards finds the deepest valley between them. The second one is an iterative method: one can choose an initial threshold, calculate a segmentation and afterwards compute a new threshold by means of segmentation until convergence is reached. Unfortunately, object and background peaks are usually not so ideally separated (due to presence of noise or poor contrast), and the choice of the threshold is complex. In fact, in this cases, the histograms are often bimodal but with peaks not well separated. Besides, if the desired goal is to partition the images in more than two parts, these approaches can be repeated recursively.

Clustering Clustering can be broadly defined as a unsupervised classification of objects in which one has to generate classes or partitions without any a priori knowledge. The problem of clustering can be analytically defined analogously to the definition of segmentation. In the

next chapter we will explain the issues related to clustering analysis and show some clustering approaches. In this thesis clustering methods are fundamental: we use a popular clustering approach for segmenting images (*k-means*) and a hierarchical algorithm (the recursive *spectral clustering*) for grouping image segments.

Chapter 3

Unsupervised and supervised learning

A powerful and accurate system for image retrieval relies on the ability to properly use the information stored in the images. Since a few decades research efforts have been made to define dissimilarity measures between image or region descriptors and underlying techniques needed to evaluate these distances.

The early approaches evaluated the power of a given image descriptor comparing its value on new images with the elements of the image dataset on the basis of similarity functions. As data sets grew large it has been observed that it was too ambitious to expect a single similarity measure to produce robust, perceptually meaningful ranking of images. As an alternative, the use of learning-based techniques has proven to have a key role in improving the effectiveness of the retrieval.

Machine learning is the field of research devoted to the formal study of learning systems. This is a highly interdisciplinary field which originates from statistics, computer science, engineering, cognitive science, optimization theory and many other disciplines of science and mathematics. In this chapter we introduce the relevant background pertaining to supervised and unsupervised learning techniques. Given the enormous range of topics under the rubric of supervised and unsupervised learning, this review is necessarily incomplete ([ROD01], [Bis06], [Ros06], and the references therein).

The remainder of this chapter is organized as follows. In Section 3.1 we discuss on the general framework of machine learning and address supervised as opposed to unsupervised learning theory. In Section 3.2 we provide a sufficient background on supervised learning paradigms and briefly introduce the regularization theory in the machine learning context. In Section 3.3 we present a brief review of the unsupervised learning approaches for setting cluster analysis in the state-of-the-art. In Section 3.4 we discuss on issues relative to popular clustering approaches and we focus on two well-known techniques that we exploited in the algorithmic

pipeline of the overall architecture system: *k-means* and *spectral clustering*.

3.1 Supervised vs. unsupervised learning

It is possible to define and distinguish between different kinds of *learning from examples*.

Assume we are provided with some *input data* $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, that could correspond, for example, to words in a text document (email or Web document), atmospheric measurements, gene expressions from DNA microarrays, or visual information contained in an image.

In *supervised learning* the input data is paired with a given a sequence of *outputs* $(y_1, \ldots, y_n)$. The idea is to infer an unknown input-output relation on the basis of the given set of input-output instances. The available data, namely the *training set*, are a collection of pairs $\mathbf{z} = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ where $\mathbf{x}$ is a vector whereas y takes on discrete or continuous values. The distinction in output type has led to a naming convention for the following prediction tasks:

- *pattern classification*, a learning problem with output values taken from a finite unordered set $C = \{C_1, \ldots, C_k\}$,
- *regression*, a learning problem whose output values are real.

If one supposes that the input and output are observations of random variables represented by some joint probability density $\rho(\mathbf{x}, y)$, than the supervised learning can be formally defined as a density estimation problem where one is concerned with determining properties of conditional density $\rho(y|\mathbf{x})$.

In *unsupervised learning* the inputs data $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ are not provided with any target outputs. In this case one has a set of n observations of a random vector $\mathbf{x}$ having a joint density $\rho(\mathbf{x})$. The goal is to directly infer the properties of this probability density without the help of a supervisor or a teacher providing correct answer or degree-of-error for each observation. In a sense, unsupervised learning can be thought of as finding patterns in the data above and beyond what would be considered pure unstructured noise.

In low-dimensional problems there are a variety of non-parametric methods Gaussian mixtures for directly estimating the density $\rho(\mathbf{x})$ but in high dimensions, because of the *curse of dimensionality*, these method fail. One must settle for estimating rather crude global methods, such as or various simple descriptive statistics attempt to characterize $\rho(\mathbf{x})$.

For example, *principal component analysis* [Pea01], *self-organizing maps* [Koh90], *multidimensional scaling* [BG05] attempt to identify low-dimensional manifolds within the space that represent high data density. This provides information about the associations among

the variables and whether or not they can be considered as functions of a smaller set of "latent" variables.

Cluster analysis attempts to find multiple convex regions of the feature space that contains modes of $\rho(\mathbf{x})$. This can tell whether or not $\rho(\mathbf{x})$ can be represented by a mixture of simpler densities representing distinct types or classes of observations.

Association rules attempt to construct simple descriptions (conjunctive rules) that describe regions of high density in the special case of very high dimensional binary-valued data.

With supervised learning there is a clear measure of success that can be used to compare the effectiveness of different methods over various situations. Lack of success is directly measured by expected loss over the joint distribution $\rho(\mathbf{x}, y)$. This can be estimated in a variety of ways including cross-validation [Efr87].

In the context of unsupervised learning, there is no such direct measure of success. One must often resort heuristic arguments not only for motivating the algorithms, as is often the case in supervised learning as well, but also for judgments as to the quality of the results. In the next sections we present an overview of the supervised and unsupervised learning theory, presenting a few of used techniques in this thesis.

3.2 Supervised learning

As we explained above, the idea of supervised learning is to infer an unknown input-output relation on the basis of a given set of input-output instances. Clearly such a task is hopeless if we do not assume some model relating input and output, that is a model for the data. Once we agree on a model, learning proceeds similarly to natural science, that is through an inference process. Then a supervised learning problem can be described as: given a certain number of observations we want to recover an approximation of the model underlying them. The problem is not trivial since we always have a finite amount of information available and various causes of uncertainty might affect the problem. In other words the problem is ill-posed [Had02], [Had23] in particular the solution is not unique and can change dramatically if we slightly change the data. In natural science one often has strong prior information which induces constraints on the possible solutions, while a main feature of learning is trying to minimize the amount of prior assumptions. Intuitively, we can say there is a trade-off between prior information and the amount of data available. Given the above premises the question is then how we can design algorithms for learning and what are the main factors determining their performances. A (simplified) graphical visualization of a 2-dimensional toy problem is useful. In Figure 3.1 each input point is a 2-dimensional vector and its label is given by its color (red or green). On the top left we see a set of data which can be thought of as a sample from a larger (possibly infinite) population on the bottom right. In this model the goal is then to draw a line such that points belonging to different classes falls in different

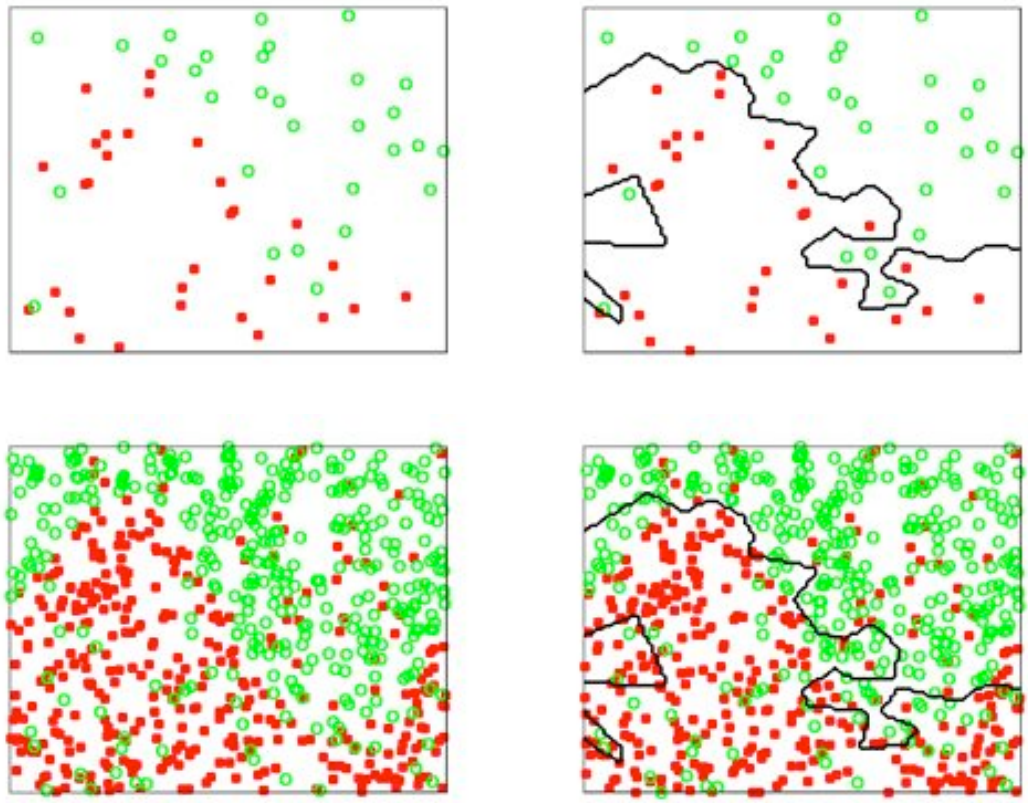


Figure 3.1: A 2-dimensional toy classification problem. Top left there is a data sample from a larger population on bottom left. Top right we can see a solution with zero empirical error and bottom right we see the poor performance of the same solution on the entire population.

sides. It is crucial to remember the goal is not only to describe the available data but rather to be predictive on new data. A solution which perfectly separates the data (top right) can perform poorly on other points of the same population (bottom right). Even in this toy model we can see some features of the problem. If we simply try to find a prediction rule which performs well on the data we tend to perform bad on new data. Clearly this is due to the fact that we have a finite number of examples. Moreover it should be clear that the more complex is the problem at hand the more examples we need: we can see an interplay between regularity of the target and number of required data.

A solution which simply describes the data is too irregular or too complex, that is it is an overfitting solution. If we postulate that the problem has some regularity properties, then we might want to impose constraints on the class of possible solutions. Regularity is described in terms of complexity (see [Bou02] and references therein) or stability of the solution (see

[PRMN04] and references therein).

In the next section we formally present some ingredients of supervised learning and regularization theory (*regularization networks* [EPP00a] or *regularized kernel methods* [CST00], [SS02], [Vap98]). We also point out the connection between algorithms in learning theory and regularization methods in inverse problems in order to introduce the main contribution of my thesis, the spectral algorithms proposed in the Chapter 5.

3.2.1 Input and output spaces

We consider two sets of random variables $\mathbf{x} \in X$ and $y \in Y$. We assume that X is a compact subset of $\mathbb{R}^d$, and the labels y_i belong into a bounded subset $Y \subset \mathbb{R}$ (for example in a binary classification problem $Y = \{-1, 1\}$). Let X and Y be related by a probabilistic relationship. The relationship is probabilistic because generally an element of X does not determine uniquely an element of Y, but rather a probability distribution on Y. As mentioned (Section 3.1), this can be formalized assuming that an unknown probability distribution $\rho(\mathbf{x}, y)$ is defined over the set $X \times Y$.

We are provided with *examples* of this probabilistic relationship, that is with a data set $\mathbf{z} = \{(\mathbf{x}_i, y_i) \in X \times Y\}_{i=1}^n$ called *training set*, obtained by sampling n times the set $X \times Y$ according to $\rho(\mathbf{x}, y) = \rho(y|\mathbf{x})\rho_X(\mathbf{x})$.

We assume that the examples $\mathbf{z}$ are drawn identically and independently distributed according to $\rho(\mathbf{x}, y)$. Moreover, we assume that X is a compact subset of $\mathbb{R}^d$, and the labels y_i belong into a bounded subset $Y \subset \mathbb{R}$ (for example in a binary classification problem $Y = \{-1, 1\}$).

Given the data set $\mathbf{z}$, the "problem of learning" consists in providing an *estimator* $f_{\mathbf{z}} : X \to Y$ that can be used to predict a value $f_{\mathbf{z}}(\mathbf{x}) \sim y$ for each $\mathbf{x} \in X$. Since we know only a finite set of points $\mathbf{z}$ the estimator $f_{\mathbf{z}}$ can be seen as an approximation of the ideal estimator $f : X \to Y$, also named the *target function*.

For example X could be the set of all possible images and Y the set $\{-1, 1\}$ which specifies whether image $\mathbf{x}$ contains a certain object and $(y = 1)$, or not $(y = -1)$. Another example is the case where $\mathbf{x}$ is a set of parameters, such as pose or facial expressions, y is a motion field relative to a particular reference image of a face.

3.2.2 Expected risk, loss functions and regression function

The standard way to deal with the learning problem consists in defining a *risk functional*, which measures the average amount of error or risk associated with an estimator, and then looking for the estimator with the lowest risk. If $\ell(y, f(\mathbf{x}))$ is the loss function measuring the error we make when we predict y by $f(\mathbf{x})$, then the average error, the so called *expected risk*,

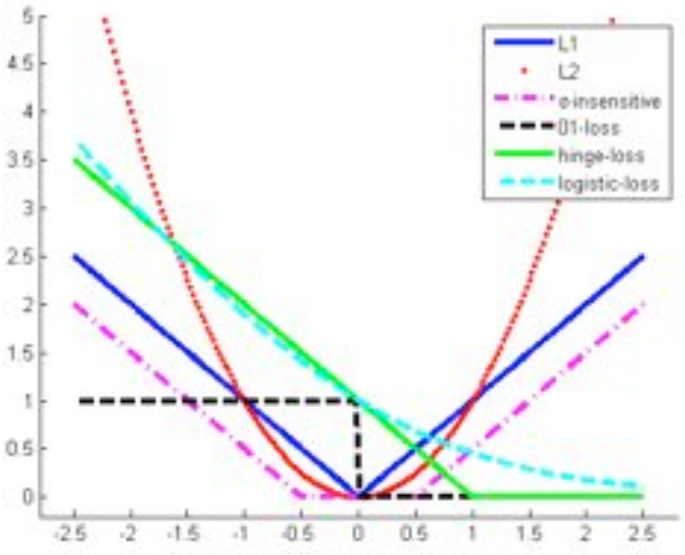


Figure 3.2: Various loss functions used in regression and classification problems.

is:

$$I[f] \equiv \int_{X,Y} \ell(y, f(\mathbf{x}))\rho(\mathbf{x}, y)\ d\mathbf{x}dy \tag{3.1}$$

Different loss functions lead to different learning algorithms [EPP00a].

Common choices are:

- the square loss $\ell(y, f(\mathbf{x})) = (f(\mathbf{x}) - y)^2$,
- the absolute value loss $\ell(y, f(\mathbf{x})) = |f(\mathbf{x}) - y|$,
- the ε-insensitive loss $\ell(y, f(\mathbf{x})) = \max\{|f(\mathbf{x}) - y|, 0\} =: |f(\mathbf{x}) - y|_\varepsilon$.

for regression, and

- the square loss $\ell(y, f(\mathbf{x})) = (f(\mathbf{x}) - y)^2 = (1 - f(\mathbf{x})y)^2$,
- the hinge loss $\ell(y, f(\mathbf{x})) = \max\{1 - f(\mathbf{x})y, 0\} =: |1 - f(\mathbf{x})y|_+$,
- the logistic loss $\ell(y, f(\mathbf{x})) = (\ln 2)^{-1} \ln(1 + e^{-f(\mathbf{x})y})$.

for classification[1].

The space $\mathcal{F}$, where the integral 3.1 is finite, is also called the target space. If we look at the expected error as a functional on $\mathcal{F}$ we can define the *target function*:

$$f_\rho = \underset{f \in \mathcal{F}}{\operatorname{argmin}} I[f]$$

which is clearly the best possible solution to the learning problem with respect to the chosen loss. The target function can be found constructively considering

$$f_\rho(\mathbf{x}) = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \int_Y \ell(y, f(\mathbf{x})) d\rho(y|\mathbf{x}).$$

We note that rather then finding functions which are similar to the target (for example similar curves) our main goal is to find some function whose expected error is close to the expected error of the target function.

One reason why the square loss can be seen as a natural choice for the loss function is that its target function is particularly easy to interpret. In fact it is possible to prove that, if

$$\int_{X \times Y} y^2 d\rho(\mathbf{x}, y) \; d\mathbf{x}dy \leq \infty$$

the expected error is defined on the space $L^2(X, \rho_X)$ of square integrable functions with respect to ρ_X and the target function is simply the expectation of the conditional probability

$$f_\rho(\mathbf{x}) = \int_Y y d\rho(y|\mathbf{x}).$$

namely the *regression function*.

3.2.3 Empirical Risk Minimization

In this section we focus on a specific learning approach, the *Empirical Risk Minimization* (ERM). The importance of ERM lies in the fact that most algorithms can be seen as refinements of it. In a few words the idea is that since we cannot minimize the expected error directly we can replace it with its empirical counterpart. Given a training set, a possible way to estimate $I[f]$ is to evaluate the *empirical risk*

$$I^{\mathbf{z}}_{emp}[f] = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(\mathbf{x}_i)).$$

[1] In the classification setting the loss function ℓ usually depends on its arguments through the product $yf(\mathbf{x})$. This particular dependency rests on the implicit assumption that false negatives ($y = +1$ and $f(\mathbf{x}) < 0$) and false positives ($y = -1$ and $f(\mathbf{x}) > 0$) are equivalent. More general situations have been considered in the literature (see for example [LLL+02]) leading to describe a loss function as $\ell(y, f(\mathbf{x})) = L(y)\ell(yf(\mathbf{x}))$.

Straightforward *minimization of the empirical risk* (ERM) is an ill posed problem, since the solution is not unique. Usually the minimization is restricted to a certain class of candidate estimators of the regression (or the target function), that is the hypothesis space. The question is then what kind of performance we can expect from this algorithm and what are the main factors affecting it. The intuition suggests that the following aspects should be relevant:

- the number of available examples n. The more examples we have the more we expect the empirical error to be a good estimator of the expected error. In general we cannot assume to have "enough" examples.
- the class of possible solutions (*hypothesis space* $\mathcal{H}$, in Section 3.2.4). On this respect we can note some interesting facts. We notice that once we choose a class of possible solutions we introduced a bias in the problem, meaning that we cannot aim at functions which are outside of the chosen space. Then we might say that we would like this space as "big" as possible. On the other hand simple examples suggest that if we have complex enough space we might incur into overfitting. Since the number of available data is limited we risk to over-estimate the information at hand loosing generalization properties. Then it looks like we should look for some intermediate "size" for the hypothesis space depending on the available data [Vap98].

3.2.4 Hypothesis space and Reproducing Kernel Hilbert Space

Before introducing *Reproducing Kernel Hilbert Space* (RKHS) we briefly illustrate the *kernel trick*.
The kernel trick is a method for using a linear classifier algorithm to solve a non-linear problem by mapping the original non-linear observations into a higher-dimensional space, where the linear classifier is subsequently evaluated. This makes a linear classification in the new space equivalent to non-linear classification in the original space. Specifically, we can project (see Figure 3.3) a data point $\mathbf{x}$ in $\mathbb{R}^d$ to a high dimensional feature space $\mathbb{R}^F$, by a nonlinear mapping function $\phi : \mathbb{R}^d \to \mathbb{R}^F$, $F > d$, and proceed to training and testing in the feature space.

The kernel trick transforms any algorithm that solely depends on the dot product between two vectors. Wherever a dot product is used, it is replaced with the kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

$$K(\mathbf{x}, \mathbf{s}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{s})$$

instead of defining $\phi(\mathbf{x})$ explicitly.
Given the training set $(\mathbf{x}_1, \dots \mathbf{x}_n)$ the $n \times n$ matrix $\mathbf{K}$ formed by $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ is called the *kernel matrix*. A natural question to ask is, given a function $K(\mathbf{x}, \mathbf{s})$ how to decide whether it is a kernel function, or whether there exists a function $\phi(\mathbf{x})$ such that $K(\mathbf{x}, \mathbf{s}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{s})$. The answer is provided by the following lemma.

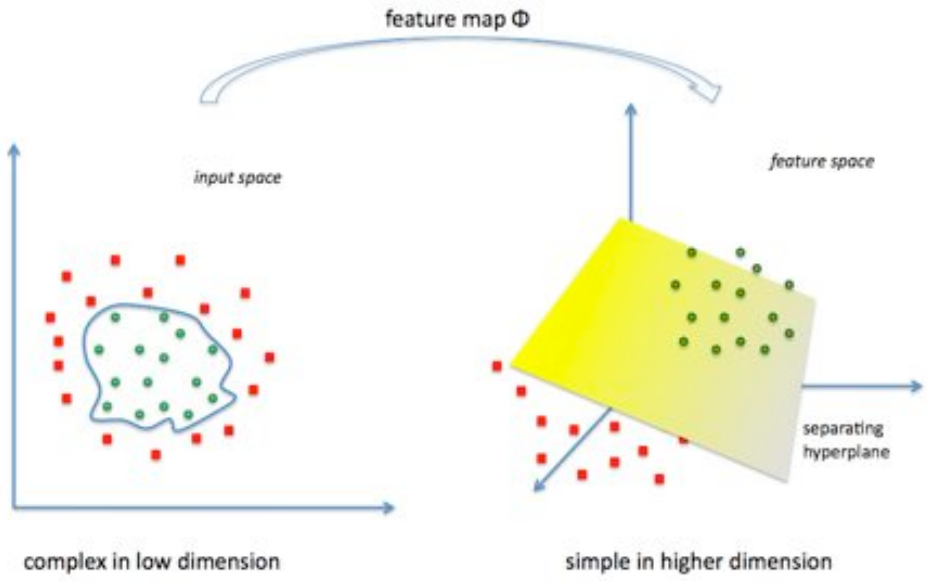


Figure 3.3: Feature mapping between input space $X \subset \mathbb{R}^d$ and $\mathbb{R}^F$.

Lemma [STC04], [SS02] A function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ can be decomposed as an inner product for some feature map $\phi(\mathbf{x})$

$$K(\mathbf{x}, \mathbf{s}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{s})$$

if and only if the function is symmetric and the matrix formed by restriction to any subset of the space $\mathbb{R}^n$ is positive semi-definite. K is a Mercer kernel if $K : X \times X \to \mathbb{R}$ is a symmetric continuous function, which is positive definite [Aro50].
An example of kernel functions is the polynomial kernel $K(\mathbf{x}, \mathbf{s}) = (\mathbf{x} \cdot \mathbf{s})^p$ where p is the polynomial degree. Another widely-used kernel that we will make use of below is the radial basis function (RBF) kernel [STC04], [Bur98]

$$K(\mathbf{x}, \mathbf{s}) = e^{-\frac{||\mathbf{x}-\mathbf{s}||^2}{\sigma^2}}$$

where σ is a *width* controlling parameter. The RBF kernel implies a function $\phi(\mathbf{x})$ that is infinite dimensional.

Predictivity is a trade-off between the information provided by training data and the complexity of the solution we are looking for. An important class of problems in regularization theory (see Section 3.2.5) are generated by a positive definite kernel $K(\mathbf{x}, \mathbf{s})$ and the corresponding space of functions $\mathcal{H}$ is called *Reproducing Kernel Hilbert Space*. A RKHS $\mathcal{H}$ on X with kernel $K : X \times X \to \mathbb{R}$ is defined as the unique Hilbert space of real valued functions

on X such that, for all $f \in \mathcal{H}$,

$$f(\mathbf{x}) = \langle f, K_{\mathbf{x}} \rangle_{\mathcal{H}} \qquad \forall \mathbf{x} \in X \tag{3.2}$$

where $K_{\mathbf{x}}$ is the function on X defined by $K_{\mathbf{x}}(\mathbf{s}) = K(\mathbf{x}, \mathbf{s})$.

In the previous sections we provided some key concepts pertaining to supervised learning problems. To prevent overfitting discussed in Section 3.2, an estimator $f_{\mathbf{z}}$ must effectively assign a label to a new point. In the next section we give an overview of theory rised to solve this problem and in Chapter 5 we provide our contribution to the state-of-the-art.

3.2.5 Regularized kernel methods

From the seminal work of Tikhonov and others [TA77] regularization has been rigorously defined in the theory of ill-posed inverse problems. In this context the problem is to invert a linear operator (or a matrix) that might have unbounded inverse (or a bad condition number). Regularization amounts to replace the original operator with a bounded operator, namely the *regularization operator* [EHN96], whose condition number is controlled by a regularization parameter. The regularization parameter should be chosen according to the noise level in order to ensure stability. Many regularization algorithms are known, Tikhonov and truncated singular value decomposition (TSVD) being probably the most commonly used.

The idea of using regularization in statistics and machine learning has been explored since a long time - see for example [Wah90], [PG92] and references therein - and the connection between large margin kernel methods such as Support Vector Machines and regularization is well known – see [Vap98], [EPP00b], [SS02] and reference therein. Ideas coming from inverse problems regarded mostly the use of Tikhonov regularization and were extended to several error measures other then the quadratic loss function.

As mentioned in Section 3.2.3, straightforward *minimization of the empirical risk* (ERM) is an ill posed problem, since the solution is not unique. The basic idea of regularization is to restore well-posedness of ERM by constraining the hypothesis space $\mathcal{H}$. A possible way to do this is considering *penalized* ERM.

We look for solutions minimizing a functional composed of two terms: the first one is a fitting term and the second is a smoothness term. This functional can be written as:

$$\underbrace{ERR(f_{\mathbf{z}})}_{empirical\ error} + \underbrace{\lambda PEN(f_{\mathbf{z}})}_{penalization\ term}$$

where λ is the regularization parameter, being a trade-off between the two terms.

We assume that the kernel K is bounded by 1 and is universal (see [MXH06] and references

therein), that is the set of functions

$$\mathcal{H} = \{\sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i) \mid \mathbf{x}_i \in X,\ \alpha_i \in \mathbb{R}\}$$

is dense in $L^2(X)$, the Hilbert space of functions that are square-integrable with respect to ρ_X.
Tikhonov regularization amounts to minimize

$$\min_{f \in \mathcal{H}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(\mathbf{x}_i)) + \lambda \|f\|_{\mathcal{H}}^2 \right\} \tag{3.3}$$

As mentioned above, the second term is a *smoothness* or a *complexity* term measuring the norm of the function f in a suitable Hilbert space $\mathcal{H}$. The minimization takes place in the *hypothesis space* $\mathcal{H}$.

The minimizer of the regularized empirical functional, for each training set $\mathbf{z} = (\mathbf{x}, y) = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$ of n-examples $(\mathbf{x}_i, y_i) \in X \times Y$, can be represented by the expression (*representer theorem*)

$$f_{\mathbf{z}}^{\lambda}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i) \quad \text{with} \quad \alpha = (\mathbf{K} + n\lambda I)^{-1}\mathbf{y}. \tag{3.4}$$

where K is a Mercer kernel and $\mathbf{K}$ is the kernel matrix.
Since $\lambda > 0$, it is clear that we are numerically stabilizing a matrix inversion problem which is possibly ill-conditioned (that is numerically unstable). With the choice of the square loss, the generalization property of the estimator means that the estimator $f_{\mathbf{z}}^{\lambda}$ is a good approximation of the regression function

$$f_\rho(\mathbf{x}) = \int_Y y \; d\rho(y|\mathbf{x}),$$

with respect to the norm of $L^2(X)$. In particular the algorithm is (weakly) consistent [Vap98] if, for a suitable choice of the parameter $\lambda = \lambda_n$ as a function of the examples,

$$\lim_{n\to\infty} \int_X (f_{\mathbf{z}}^{\lambda}(\mathbf{x}) - f_\rho(\mathbf{x}))^2 \; d\rho_X(\mathbf{x}) = \lim_{n\to\infty} \left\| f_{\mathbf{z}}^{\lambda} - f_\rho \right\|_\rho^2 = 0$$

with high probability – see for example [Vap98].

As mentioned, empirical risk minimization is an ill-posed problem and the formulation of regularization comes from inverse problems. A large amount of literature had pointed out the connection between algorithms in learning theory and regularization methods in inverse problems, see for example [Vap82, PG92, Vap98, EPP00b, HTF01, SS02, DVRC$^+$05]. The main message is that regularization techniques provide stability with respect to noise and sampling, therefore ensuring good generalization properties to the corresponding learning algorithms.

3.2.6 Consistency

In the previous section we argued that a possible way to state the goal of learning is to say that we are looking for an estimator with small expected error. Here we formalize such a statement.
We have seen that each loss function induces a target function which can be seen as the best possible solution to the learning problem since it achieves the minimal error. It is then natural to look at the the deviation $I[f_{\mathbf{z}}] - I[f_\rho]$, sometimes called *excess error*, to measure the generalization property of our estimator. Considering a worst case scenario, the analysis amounts to study probabilistic inequalities of the form

$$Pr(I[f_{\mathbf{z}}] - I[f_\rho] \geq \varepsilon) \leq \eta(\varepsilon, n)$$

where $\varepsilon > 0$ and $n \in \mathbb{N}$. We say that an estimator $f_{\mathbf{z}}$ (or the corresponding learning algorithm) is *consistent* if

$$\lim_{n\to\infty} Pr(I[f_{\mathbf{z}}] - [f_\rho] \geq \varepsilon) = 0$$

Though the above statements are asymptotic they can be seen as minimal sanity checks ensuring that the learning algorithm performs better as more data are available and eventually leads to the best attainable solution with an infinite number of samples.

In Chapter 5 we will show that regularized least squares, as well as a large class of regularization methods, can be seen as a suitable filtering of the kernel matrix, able to ensure good generalization properties of the estimator. The proposed methods have many interesting features: from the algorithmic point of view they are simple to implement, usually they amount to a few lines of code. They are appealing for many applications: their model selection is simple since they depend on few parameters, and over-fitting may be dealt in a very transparent way.

3.3 Unsupervised Learning

The purpose of this section is to introduce in a fairly concise manner the key ideas underlying the sub-field of machine learning known as unsupervised learning. This introduction is necessarily incomplete given the enormous range of topics under the rubric of unsupervised learning.

As mentioned above, we will focus on the main algorithms of clustering used in the unsupervised learning. However, in this section we try to provide a rough taxonomy of the most commonly used unsupervised learning techniques to introduce clustering analysis in the general framework of unsupervised learning:

- *Association Rules*: Let $\mathbf{x} = (x^1, \ldots, x^d)$ a set of attributes called items, the main goal of the association rules analysis is to find a collection of prototypes $\mathbf{t}_1, \ldots, \mathbf{t}_L$ for the

feature vector $\mathbf{x}^j$, such that the probability density $\rho(\mathbf{t}_\ell)$ with $\ell = 1, \ldots, L$ is relatively large.

When dealing with binary-valued data $x^j \in \{0, 1\}$, also known as "market basket" analysis, this corresponds to "mode finding" or "bump hunting".

Indeed, in commercial databases, the observations are sales transactions, such as those occurring at the checkout counter of a store. The variables represent all the d items sold in a store: for the customer i, $x_i^j = 1$ if the j-th item is purchased, 0 otherwise, with $j = 1, \ldots, d$. This information can be quite useful for stocking shelves, cross-marketing in sales promotions, catalog design, and the consumer segmentation based on buying patterns.
When analyzing the data we assume that the basket content behaves accordingly to a certain probability distribution. In general solving the problems related to market basket analysis with a naive algorithm and a very large dataset is computationally expensive.

The "Apriori" algorithm [AIS93] exploits several aspects of the curse of dimensionality to solve, with a small number of passes, a simplified problem of the one above cited. Apriori, while historically significant, suffers from a number of inefficiencies or trade-offs, which have spawned other algorithms.

- *Dimensionality reduction*: The problem of dimension reduction is introduced as a way to overcome the curse of the dimensionality when dealing with vector data in high-dimensional spaces and as a modeling tool for such data. It corresponds to the search for a low-dimensional manifold that embeds the high-dimensional data. The dimensionality reduction problem consists of extracting low dimensional data structure from high dimensional data. The problem can be stated as follow: given the d-dimensional random variable $\mathbf{x} = (x^1, \ldots, x^d)$, find a lower dimensional representation of it, $s = (s_1, \ldots, s_k)$ with $k \leq d$, that captures the content in the original data, according to some criterion. The traditional state-of-the-art of dimension reduction methods covers with *principal components analysis* (PCA) [Pea01], *projection pursuit* [FT74], *principal curves* [Has84], *self-organizing maps* [Koh90], *independent component analysis* [Com94], *multidimensional scaling* and [BG05]. New techniques include ISOMAP [TdSL00], *locally linear embedding* (LLE) [RS00], *Laplacian* [BN02], *Diffusion Maps* [CL06], etc. For surveys or books in the statistical and machine learning literature on these techniques see for example [HTF01].

- *Cluster analysis*: clustering is the assignment of objects into groups (called *clusters*) so that objects from the same cluster are more similar to each other than objects from different clusters. In the next section we briefly describe cluster analysis and related issues, finally we show two well-known clustering techniques: *k-means* [Mac67] and *spectral clustering* [SM00].

3.4 Clustering Analysis

Cluster analysis, also called data segmentation has a variety of goals. All of them relate to grouping or segmenting a collection of objects into subsets or clusters such that those within each cluster are more closely related to one another than objects assigned to different clusters. An object can be describes by a set of measurements, or by its relation to other objects.
In addition the goal is sometimes to arrange the clusters into a natural hierarchy. This involves successively grouping the clusters themselves so that at each level of hierarchy, clusters within the same group are more similar to each other than those in different groups.
Cluster analysis is also used to form descriptive statistics to ascertain whether or not the data consists of a set distinct subgroups, each group requires an assessment of the degree of difference between the objects assigned to the respective clusters. Central to all of the goals of cluster analysis is the notion of the degree of *similarity* (or dissimilarity) between the individual objects being clustered.

A clustering method attempts to group the objects based on the definition of similarity supplied to it. The situation is somewhat similar to the specification of a loss function (see Section 3.2.2) in prediction problem (supervised learning).
In addition to similarity/dissimilarity measures, another important issue is determining how many classes the data objects should be grouped into. As mentioned above, clustering is unsupervised. Hence, the number of clusters in addition to the class labels of the data objects, is unknown. A clustering algorithm can be used to infer the class labels. However, for most clustering algorithms, the number of clusters has to be given a priori. Unfortunately, there is no general agreed upon way to find the optimal number of clusters for any given data set.

We first discuss distance measures before describing algorithms for clustering.

3.4.1 Similarity measures

The most common approach normally first considered in clustering is to simply use (squared) Euclidean distance $d(\mathbf{x}, \mathbf{s}) = ||\mathbf{x} - \mathbf{s}||^2$ to measure the dissimilarity between data objects. The choice of squared Euclidean distance implies that the resulting clusters will be invariant to translations or rotations. However, they will not be invariant to transformations that distort distance relationships. For example, different axis scalings can lead to different clusterings. One way to avoid this problem is to normalize the data before clustering, which means making each attribute of the data has zero mean and unit variance. In general, a Mahalanobis distance could be used, where $d(\mathbf{x}, \mathbf{s}) = (\mathbf{x} - \mathbf{s})^T \Sigma^{-1} (\mathbf{x} - \mathbf{s})$ and Σ is an arbitrary positive definite covariance matrix that could be estimated from data [Mah36]. However, naive rescalings and estimates of Σ might be inappropriate if the large variance of an attribute is due to the presence of clusters. In this case, domain knowledge will be important to identify which

attributes have higher influence in defining object similarity.
Instead of defining distance measures directly, another common approach is to define a *similarity* measure $s(\mathbf{x}, \mathbf{s})$ from which one can derive a distance. For example, the normalized inner product

$$s = \frac{\mathbf{x}^T \mathbf{s}}{||\mathbf{x}||\ ||\mathbf{s}||}$$

which gives the cosine of the angle between $\mathbf{x}$ and $\mathbf{s}$, is a popular choice to define the similarity of meaning in natural language. A similarity measure like this is similar to a kernel, as defined in Section 3.2.4. In fact, the cosine similarity corresponds to an inner product between feature vectors

$$s(\mathbf{x}, \mathbf{s}) = \frac{\mathbf{x}}{||\mathbf{x}||} \cdot \frac{\mathbf{s}}{||\mathbf{s}||}$$

and thus by Lemma (see Section 3.2.4) is a kernel. Standard kernels can therefore be used as similarity measures for clustering. For example, beyond the cosine function, the RBF kernel introduced previously (see Section 3.2.4) in the context of supervised learning is also a popular choice for a similarity measure in clustering algorithms.

3.4.2 Clustering algorithms

The clustering problem can be stated as follows: let us suppose that we have n patterns $\mathbf{z} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ within a certain space X. Clustering consists in determining groups $C_1, \ldots, C_K$ such that every $\mathbf{x}_i$, $i = 1, \ldots, n$, belongs to one of these groups and no $\mathbf{x}_i$ belongs to more than one group at a time, i.e., $\bigcup_{k=1}^{K} C_k = \mathbf{z}$ and $C_h \cap C_s = \phi$ with $h, s = 1, \ldots, K$. The classification of patterns into classes follows the general common sense principle that objects within each class should show a high degree of similarity while across different classes they should exhibit very low affinity.

Different starting points and criteria usually lead to different taxonomies of clustering algorithms (see [JD88], [ELL01] and [Kol01]). A rough but widely agreed frame is to classify clustering techniques as *hierarchical clustering* and *partitional clustering*, based on the properties of clusters generated.

Partitional clustering directly divides data objects into some pre-specified number of clusters without the hierarchical structure.

Hierarchical clustering groups data objects with a sequence of partitions. Strategies for hierarchical clustering divide into two basic paradigms: *agglomerative* and *divisive*.
Agglomerative strategies proceed bottom-up and each level recursively merge a selected pair of clusters into a single cluster. The pair chosen for merging consists of the two groups with the smallest intergroup dissimilarity.
Divisive methods start at the top and each level recursively split one of the existing clusters

at that level into two new clusters. The split is chosen to produce new groups with the largest between-group dissimilarity.

In the next sections, we will mainly focus on the *k-means* [Mac67] clustering and *spectral clustering* [SM00] methods. The spectral clustering, derived from the graph theory, belongs to partitional clustering algorithms. However, we will focus on the recursive formulation of spectral clustering technique proposed in [SM00], that we implemented and exploited for thesis purposes.

K-means clustering

K-means is still one of the most popular clustering algorithms in use today for clustering data. As mentioned above, the method belongs to the class of *partional clustering*: it consists of dividing objects without any hierarchical structure on clusters. This method classifies a given data set into subsets by an iterative minimization of a within-class dissimilarity measure. For k-means clustering, squared Euclidean distance is normally chosen as the dissimilarity measure, although, in general, any distance measure could be used. Given a set of unlabeled data points $(\mathbf{x}_1, \dots \mathbf{x}_n)$, the main idea of k-means is to define K centers, $\mathbf{c}_1, \dots \mathbf{c}_K$, one for each cluster, with the goal of minimizing the sum of distances to the class centers to which data points are assigned. That is, the algorithm aims at minimizing the following objective function:

$$J = \sum_{k=1}^{K} \sum_{i=1}^{n} \|\mathbf{x}_i^{(k)} - \mathbf{c}_j\|^2$$

where $\|\mathbf{x}_i^{(k)} - \mathbf{c}_j\|$ is the Euclidean distance between a data point $\mathbf{x}_i^{(k)}$ and the cluster centre $\mathbf{c}_j$ and K is the number of clusters.

K-means is an iterative algorithm, organized in the following steps:

1. Initialize the algorithm choosing randomly K points in the feature space: these points represent the initial centroids of the cluster

2. Assign each object to the group that has the closest centroid.

3. When all data points have been assigned, recalculate the positions of the K centroids.

4. Repeat steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

Although it can be proved that the procedure will always terminate, the k-means algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. The popularity of the k-means algorithm is due to its low computational

complexity of $O(nktd)$, where n is the number of data points, d is the dimension of the space, and t the number of iterations which is always related to n.

The algorithm is also significantly sensitive to the initial randomly selected cluster centers: as mentioned, k-means is not guaranteed to return a global optimum. Since the algorithm is extremely fast, a common method is to run the algorithm several times and return the best clustering found. Another drawback of the k-means algorithm is that requires a reasonable guess for the number of clusters present. An inappropriate choice of K may yield poor results.

Spectral Clustering

Another well-known approach was proposed by Shi and Malik [SM00]. They tackle clustering as a graph partitioning problem.

They represent the set of points in an arbitrary feature space as a weighted undirected graph $G = (V, E)$, where the nodes are the points in the chosen feature space and edges are established between each pair of nodes. The weight $w(i, j)$ of each edge is a function of the similarity between nodes i and j. The goal is to partition the set of vertexes V into disjoint sets $V_1, \ldots, V_m$ such that a predefined similarity measure is high for vertices's within the same set and low across different sets. Therefore they proposed to minimize the following measure also named *Normalized Cut*:

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

where $cut(A, B)$ is the sum of the weights between A end B and $assoc(A, V)$ are the total connections from each node in A to all nodes in the graph. For background details refer to [SM00].

Rewriting the normalized cut problem and relaxing the solution, the final expression can be found by solving the generalized eigenvalue system:

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

where D is the diagonal matrix of the total connections for each node of the graph.

There are two different versions of normalized spectral clustering proposed in [SM00], the k-way and the *recursive two-way* algorithm. One technique is to use multiple eigenvectors simultaneously: the idea is to first embed the data points in the space of the top k eigenvectors, and then cluster the embedded points using another clustering algorithm, such as k-means 3.4.2.
An alternative is to perform recursive clustering, which requires repetitions of the eigenvector computation but does not require a prior-knowledge on the number of clusters.

The steps of the recursive algorithm are:

1. define similarity measure between two nodes of the graph and build the weight matrix W,
2. solve the eigenvalue system $D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}\mathbf{x} = \lambda\mathbf{x}$,
3. use the eigenvector with the second smallest eigenvalue to bipartite the graph,
4. decide if the current partition should be subdivided imposing a threshold on $Ncut$ value and repeat the procedure on the sub-clusters if necessary.

Shi and Malik [SM00] show an application of this algorithm to an image segmentation problem. Since they just use the brightness values of the pixels and their spatial location, they define the graph edge weight connecting two nodes by means of the multiplication between the gaussian kernel of the brightness values and gaussian kernel of the spatial coordinates. They also impose a stability criterion on the partition depending on the segmentation scheme [SM00]. In Chapter 4 we will give further details on the recursive method and, we present our choice of the weight matrix and the stopping procedure.

Chapter 4

Discovering concepts from tagged images

In this chapter we describe the proposed algorithmic methodology to infer the significant *semantic concepts* represented in the images. The concepts are necessary to define the database of labels (or tags) used for the classification and annotation of image regions.
In our approach, each image is first segmented into a number of homogeneous sub-regions and its visual information is encoded by means of suitable features. Consequently, the *annotation process* consists in associating one or more labels to the images on the basis of the visual appearance of their sub-regions, while – given a keyword – the *retrieval* depends on presence in the image of at least one subregion labeled with such keyword.
Since the automatic *annotation tool* and the *search engine* are implemented through a pool of *supervised classifiers*, a crucial issue to address is the construction of appropriate datasets to train the statistical learning modules.
The specific aim of this chapter is to describe how to build the training and validation sets for the subsequent stages of the analysis by means of an unsupervised categorization of all the blobs extracted from the images.

The chapter is organized as follows. In Section 4.1 we present the system overview describing briefly the first steps towards to the annotation ad retrieval system. In Section 4.2 we describe how the input images are preprocessed by applying an effective segmentation algorithm based on color-texture features. In Section 4.3 we introduce the descriptors for representing the image regions and discuss how to cluster the blobs in order to discover meaningful subgroups that can be likely associated to semantic concepts. In Section 4.4, we present the methodology conceived to assign to each blob a semantic concept.

4.1 The algorithmic pipeline

The proposed system is quite articulated and comprises modules which are conceptually very different from each other. For this reason we first provide an overall glimpse of the entire pipeline and then we focus on each module separately. Figure 4.1 depicts the links between the modules of the system which are mainly involved in the training stage, where we build the statistical models of the semantical concepts.

In order to design and build the system, we had to address three key questions. First, *what might we use as the fundamental bricks in our CBIR system*? We questioned whether images themselves has to be used necessarily as the basic query elements or if it was worthwhile indeed

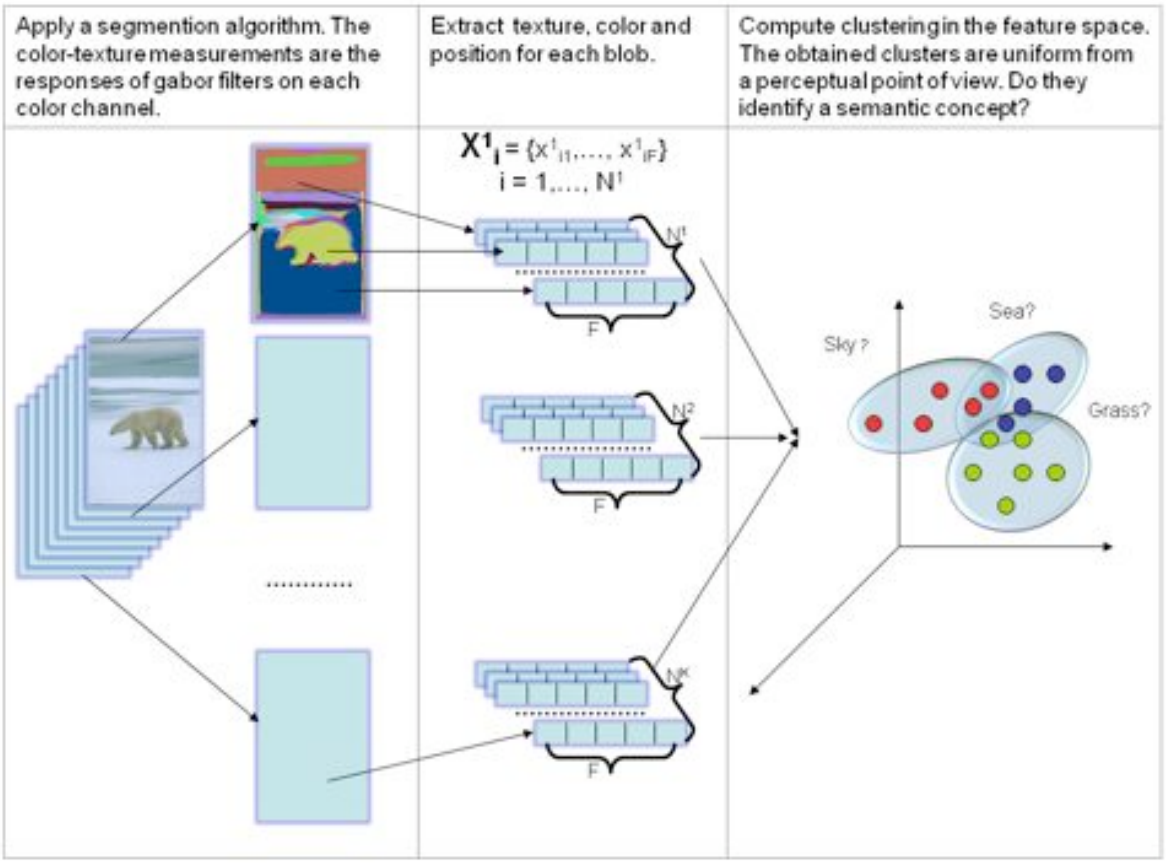


Figure 4.1: Overview of approach for the algorithmic definition of a *visual concept*. The images (on the left) first undergo a segmentation process based on color and texture simultaneously. The resulting image parts are clustered and then some of the obtained subsets of blobs are associated to significant concepts. More details are in the remainder of this section.

to refine the analysis by considering more localized information within each image. In this latter case it would have been crucial to balance between the granularity of the description and the computational overload of the consequent analysis. Second, *how can we translate into actual and procedural rules the vague idea of semantic concept associated to an image*? From an algorithmic point of view, we had to formalize syntactically the expression "semantic concept" and learn to discriminate between different concepts. Third, *how do we learn from data in a computationally efficient manner all relevant concepts?* As opposed to the fully supervised approach in which the human designer decides which is important beforehand, our opinion is that the exploration and exploitation of the regularities (i.e. the common patterns) present in the data should make the system scalable and likely to adapt itself to deal with new concepts. Actually, one obvious drawback of such viewpoint is that sometimes semantic concepts are intimately connected to the subjective interpretation of an human user and to its ability to relate multiple sources of contextual information – our system would fail short to capture such concepts. Nonetheless, preliminary experiments show that a number of concepts, which play an important role in the usual semantic querying scenario, can be dealt successfully by the proposed system.

Our specific answer to the first question is to focus on homogeneous image regions that have an high probability to be representative of a specific object or a coherent region of interest present in the scene. We extract homogeneous regions by segmenting the images using textural and color cues. In order to make a quantitative analysis of the regions we associate a fixed feature vector to them, which summarizes not only textural and color-related information but also the absolute position of the region in the image. Since this pre-processing stage cannot be too computationally expensive we adopt a fairly coarse segmentation approach, which is robust enough but not very accurate in retrieving correctly shape information of the region. Consequently, we do not use the analysis of the shape to form the feature vector of the obtained regions.

In order to define "procedurally" the notion of semantic concept associated to homogenous regions of images we rely on a discriminative approach by building automatically linear classifiers, each trained to recognize those image parts that are instances of a specific concept. Our proposal is alternative to a *top-down* explicit definition of *what* each semantic concept by building its probability model in a fully supervised way.
In consequence of the previous choices, the most difficult part in developing the system is to learn from data both the *notion of relevant concept* and the *statistical estimator* that links each image part to the correct concept. Our original contribution to this problem is the design of an unsupervised procedure, based on the spectral clustering algorithm, that builds up such notion directly from the data, by exploiting the *geometrical information* contained in the feature space associated to all the image parts extracted from all the available images. The statistical models for discriminating the concepts are, therefore, defined for each concept by means of an effective spectral algorithm recently proposed in [LGRO$^+$08] and discussed in Chapter 5.

To summarize, the proposed approach to CBIR is based on an *architecture of classifiers*, each responsible for a specific semantic concept. Given an input image $\mathbf{I}$, we first segment it coarsely obtaining an arbitrary number of *segments* $\{B_1, \ldots, B_n\}$, and assign a meaningful label ℓ_i to each B_i $(i = 1, \ldots, n)$ on the basis of the output of the classifiers. The *semantic* class of the image is obtained by combining all the ℓ_i. Henceforth we call B_i^j the i-th *blob* associated to the image $\mathbf{I}_j$; when no ambiguity may arise, we omit to repeat the index j in order to ease the notation. The modules on which our system is based are the following.

1. *Image-To-Blobs Decomposition*, in which we preprocess the input images applying a simple segmentation algorithm based on color-texture features.

2. *Unsupervised Categorization of Image Blobs*, in which we first compute a feature description for each blob using global features, and then cluster the blobs in order to discover meaningful subgroups that can be likely associated to semantic concepts.

3. *Supervised Learning of Conceptual Classes*, in which we build a binary classifier to discriminate each relevant cluster discovered in the previous step. The classifier is used as *search engine* to assign a semantic label to the blobs of each query image.

Once all the classifiers for the conceptual classes are built, they can be used to create a hierarchical architecture to associate a semantic label to all the blobs of an image and, consequently, a vector of labels to all the images in a repository for subsequent *content-based retrieval*. Moreover, the system can be used to analyze new images and assess if they contains or not a specific visual concept; which corresponds to *automatic annotation*.
In Section 6.2 we show that this part of the system can be actually exploited to allow the users to make queries and retrieval based on *visual tokens*, within a subset of the popular COREL30K dataset.

4.2 Image-to-blobs decomposition

In this first stage of the analysis our aim is to identify the blobs, defined as homogeneous subregions of the image. Our approach to such this problem is an adaptation of the framework proposed by [HGS05] for appearance-based segmentation. From the algorithmic point of view, the framework integrates the measurement of color and texture in combination – rather than color or texture alone – by fusing the output of traditional Gabor filters [Gab46] and Gaussian derivative filters [Jul81] within a specialized color space, in which the spatial frequency is measured by sampling images with a shifted Gaussian in the spatial frequency domain, while the color is measured by sampling the signal with Gaussian in wavelength domain. Our experiments confirmed that this representation schema provide effective discriminating power yielding good results.

As for many natural visual systems, the process of image formation in computer vision begins with the light rays which enter the camera and hit the *image plane*, the camera's photosensitive device which registers light intensities.
An essential part of the image formation is the *radiometry*, concerning with the relation among the amounts of light energy emitted from the light source ands, reflected from the surfaces, and registered by sensors. Therefore, before observation, a color image may be regarded as a three-dimensional energy density function $\mathbf{I}(x, y, \lambda)$ where (x, y) denotes the spatial coordinates and λ denotes the wavelength. Observation of the energy density $\mathbf{I}(x, y, \lambda)$ boils down to correlation of the incoming signal with a measurement probe $p(x, y, \lambda)$:

$$\hat{M}(x, y, \lambda) = \int\int\int \mathbf{I}(x, y, \lambda) p(x, y, \lambda) dx dy d\lambda$$

If we have a spatially shift invariant imaging system and consider the Gaussian shape as a probe function, we write the color-texture measurement as:

$$\hat{M}(x, y, \lambda) = h(x, y) * \hat{\mathbf{I}}_{\lambda^{(n)}}$$

where

$$\hat{\mathbf{I}}_{\lambda^{(n)}} = \int \mathbf{I}(x, y, \lambda) G_n(\lambda - \lambda_0; \sigma_\lambda) d\lambda$$

is the color measurement of $\mathbf{I}(x, y, \lambda)$ obtained by sampling with a n-th order Gaussian derivative,
and

$$h(x, y) \propto \exp\left(\frac{-x^2 + y^2}{2\sigma_s^2}\right) \cdot \exp\left(2\pi j(Ux + Vy)\right)$$

is the 2D Gabor function at the radial center frequency $F = \sqrt{U^2 + V^2}$ and the filter orientation $\tan(\theta) = V/U$.

More importantly, the measurements correspond well with the opponent color theory for human vision. Here, the first-order Gaussian derivative probe compares the blue region at the color spectrum with the yellow part, whereas the second order measurement compares the middle green part with the outer (magenta) regions.
In [GvdBSG01] they show that $(\hat{\mathbf{I}}, \hat{\mathbf{I}}_\lambda, \hat{\mathbf{I}}_{\lambda\lambda})$ can be approximate by the following linear transformation:

$$\begin{pmatrix} \hat{\mathbf{I}} \\ \hat{\mathbf{I}}_\lambda \\ \hat{\mathbf{I}}_{\lambda\lambda} \end{pmatrix} = \begin{pmatrix} 0.06 & 0.63 & 0.31 \\ 0.19 & 0.18 & -0.37 \\ 0.22 & -0.44 & 0.06 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

(a) Input (b) $\hat{\mathbf{I}}$ (c) $\hat{\mathbf{I}}_\lambda$ (d) $\hat{\mathbf{I}}_{\lambda\lambda}$

Figure 4.2: An example of the three channels of color gaussian model. Figure 4.2(b) represents the lightness and 4.2(c), 4.2(d) indicate respectively red/green and yellow/blue spectrum.

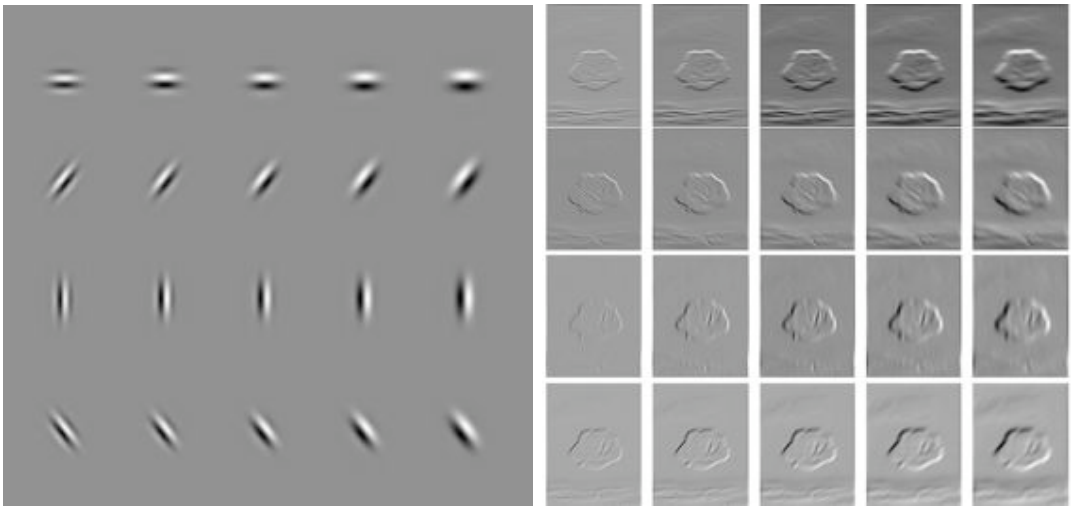
Figure 4.2 shows an example of the three color channels of the gaussian color model. We can notice that the appearance of the three images is very similar to the one shown for the CIE $L^*a^*b^*$ space in Figure 2.4.

The selection of the optimal filters to be used in order to build the color-texture representation is performed according to the guidelines proposed by [JF90] and [Man96]. They model the *multi-channel* representation with a fixed set of gabor filters that preserve almost all the information in the input image. This filter set constitutes an approximate basis for a wavelet transform, with the gabor filter as wavelet. The decomposition obtained is nearly orthogonal, as the amount of overlap between filters (in the spatial frequency domain) is small. Therefore, filtering process is adaptive to the image dimension and eliminates redundancy in the representation. In our experiments, we use 20 Gabor filters built from five scales corresponding to five center f_0 frequencies from 0.05 to 0.20 (cycles/pixel), and four orientations $\theta = 0, \pm\pi/4, \pi/2$. Imaginary part of the chosen bank of filters and corresponding responses are shown in figure 4.3.

The typical texture feature extraction process does not consist exclusively of the calculation of the filters responses [RH99]. First, since gabor filters are constituted from an imaginary and a real part, the magnitude of the two image responses is computed. The magnitude response emphasizes texture regions, which are in tune with the chosen frequencies of the filter. Second, a locally energy function is applied to the filter responses:

$$\psi(x) = tanh(\alpha x) = \frac{1 - e^{-2\alpha x}}{1 + e^{-2\alpha x}}.$$

The local energy function represents a non-linear transformation that can be interpreted as a texture-region detector. Finally the image is filtered with a separable gaussian filter (depending on the radial center frequency):



(a) Features (b) Responses

Figure 4.3: We used the set of Gabor filters whose imaginary components are shown in the left part of the image. The image responses over all filters for the first color channel ($\hat{\mathbf{I}}$) are shown in the right part.

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}},$$

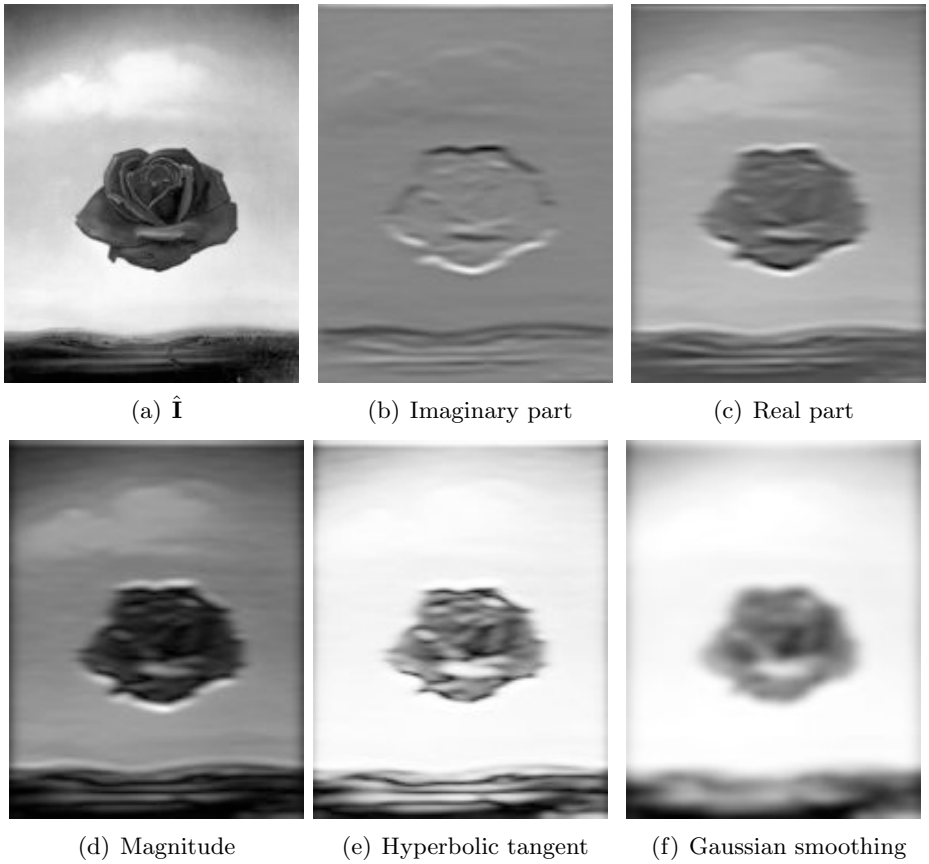
where

$$\sigma = \frac{1}{2\sqrt{2}f_0}.$$

Figure 4.4 shows the above steps in order to extract representative texture features.

We therefore obtain 60 filtered response images from which we consider the magnitude $R_d(x, y)$, $d = 1, \ldots, 60$. Each image pixel (x_i, y_i) is now represented by a 60-dimensional feature vector whose d-th component is denoted by $R_d(x, y)$. Pixels for each color-texture homogeneous region will form a cluster in the feature space, which is compact and may be discriminated from clusters corresponding to other regions.

Since we are not interested in the *optimal* segmentation but we aim at creating quickly coarse blobs for the further stages of the analysis we modified the segmentation algorithm by eliminating the post-processing stage: we do not perform shadow invariance, removal of small isolated regions and PCA analysis. Instead, after a first clustering process in which we separate homogeneous regions in the feature space, we introduced a refinement in which we compute the connected components into the image plane in order to obtain clusters that are homogeneous in both feature and spatial domains.
The segmentation results are illustrated in Figure 4.5 and Figure 4.6. The results show



(a) $\hat{\mathbf{I}}$ (b) Imaginary part (c) Real part

(d) Magnitude (e) Hyperbolic tangent (f) Gaussian smoothing

Figure 4.4: Illustration of the texture feature extraction process: 4.4(a) first channel of the color gaussian model, 4.4(b) response to the imaginary filter with the orientation $\theta = 0$ and size 31×31 ($\sigma = 5$) 4.4(c) response to the real part, 4.4(d) magnitude 4.4(e) hyperbolic tangent, 4.4(f) smoothing.

that regions are correctly discriminated (if we consider a weak-segmentation) when using the color-texture measurements described above.

Summarizing the key steps of the algorithm, we have:

1. compute the linear transform from RGB to Gaussian color model $(\hat{\mathbf{I}}, \hat{\mathbf{I}}_\lambda, \hat{\mathbf{I}}_{\lambda\lambda})$,
2. apply Gabor filters (see Figure 4.4(a)) to each channel and compute the corresponding features according to the schema proposed by [RH99],
3. apply k-means algorithm, using a fixed k chosen according to some heuristic during the designing phase of the system and
4. compute connected regions from clusters.

(a) Input (b) Segmentation (c) Blobs

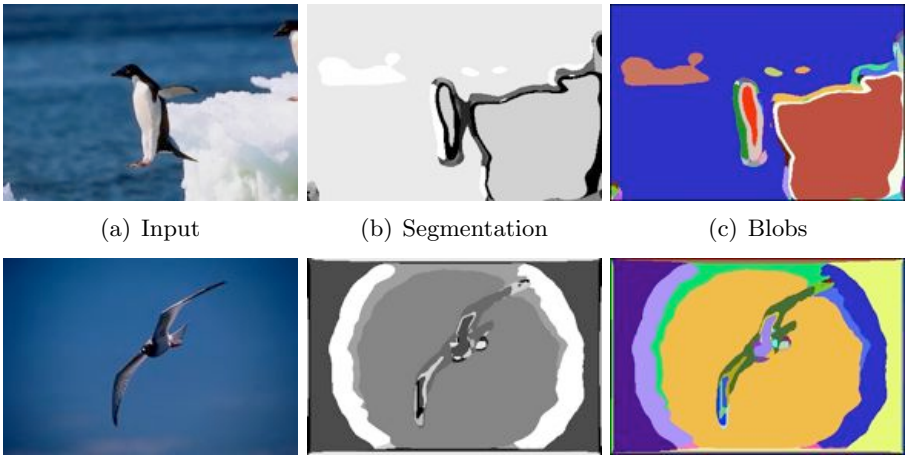
Figure 4.5: Output of the segmentation of a test image by our system. The pixels in the original image are first segmented in the feature space – obtaining possibly disconnected segments in the image plane – and then are clustered according to their spatial coordinates in order to obtain blobs.

The algorithm has several tunable parameters: the parameters of Gabor filters, the parameter of the nonlinear transformation, the band of smoothing and the k-value of k-means algorithm. The filter parameters, the nonlinear transformation and the σ of the gaussian are chosen automatically and depend on the size of the input image. The k number is determined by means of an heuristic ($\simeq 10$) since that the connected component algorithm yields hundreds of blobs for each image. Actually, we are interested in the blobs of major dimension, hence we remove meaningless blobs through a threshold value on the area size before of clustering regions. This is a significant aspect of the our approach: each phase of the pipeline proceed automatically, for each kind of provided input.

4.3 Unsupervised categorization of blobs

This stage is central in our approach since here we try to discover patterns of similar blobs, deriving from them some notion of semantic concept. Specifically, we first cluster together the features vectors representative of each blob by taking into account color, texture and relative position within the image plane. and then we designed a simple unsupervised procedure to *extract* – if possible – a candidate label to be the semantic concept encoded by the visual appearance of all the blobs in the cluster. The approach is entirely unsupervised and assumes there is a set of representative images from which to build the model. At the beginning of the next section we discuss the choices made to build the training set.

The visual properties of a blob are summarized by using of textural and color-related information, while geometrical information is solely encoded by the absolute position of the region



(a) Input (b) Segmentation (c) Blobs

(d) Input (e) Segmentation (f) Blobs

Figure 4.6: Segmentations of natural images from Corel database

in the image. In fact, we prefer a slightly coarse segmentation approach that is computationally affordable and robust enough also in presence of complex visual patterns, but not necessarily accurate in retrieving correctly shape information of the region. Consequently, we can not make the analysis of the shape part of the feature vector for the segmented regions. In addition, it is worth pointing out that most of the objects typically present in outdoor images have not a well-defined shape and therefore is such cases the boundary can not be representative of that specific concept. This is a further motivation that led us to disregard explicit representation of shape information in this first version of the system.

To summarise, for each blob we compute:

- the distribution of the pixels in the blob after the conversion of the image to the Hue-Saturation-Value (HSV) color space. In order to avoid some well-known pitfalls related to HSV histograms we discard the V component and use quantization of the other two channels. The total number of bins is 50; 10 for the H and 5 for the S channels.

- the mean and the standard deviation of Gabor filter responses already used in the segmentation stage. Therefore, this adds another 40 components to the feature vector.

- the position of the centroid of the blob in the image plane, using normalized dimensions relative to the number of columns and rows.

After the previous feature computations, each blob B_i is represented by a feature vector in a high-dimensional space, and our next step is to cluster the set of all such vectors by means of an adapted implementation of popular spectral clustering method.

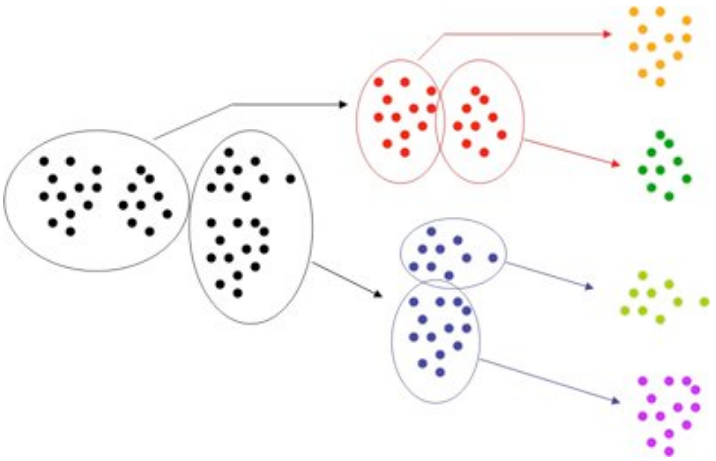


Figure 4.7: Schematic description of the hierarchical bi-partitioning produced by our recursive spectral clustering algorithm. The algorithms has been used to discover patterns within the set of all the blobs extracted by the training images from the COREL30K dataset.

4.3.1 The clustering algorithm

Spectral clustering refers to a class of techniques to partition points into disjoint clusters by analyzing the eigenstructure of the Laplacian matrix of a similarity graph associated to the dataset. In [Chu97], the first few eigenvectors of the graph's Laplacian are shown to carry information about the optimal cut to partition the graph. Different algorithms have been proposed in the last years that exploit this result (see [NJW02] and [SM00] for two popular examples). We implemented an efficient recursive algorithm for spectral clustering following the guidelines of [SM00] (see figure 4.7 for a simple schematization of the resulting hierarchical bi-partitioning approach). Indeed, the following features of such algorithm make it a valuable tool in our context. First, we do not have to decide the number of the clusters beforehand, instead we must define only the minimum size allowed for the small clusters. The level of granularity is specified through a threshold on the value of the cut. The partitioning is stable across multiple runs of the algorithm because we do not need a final k-means step as required by algorithms derived from that proposed by [NJW02]).

Despite the recursive version of spectral clustering is not new in the literature and its effectiveness has been acknowledged by several researchers, so far most of the implementation work has been focusing on the non recursive version. Therefore, we believe the general design and efficient implementation of the clustering method can be considered a valid contribution to the state of the art in this field, and it is worth specifying some of the main feature of the implemented package.
From an algorithmic point of view, the rationale of the recursive method is to build a binary tree whose *root* represents the whole dataset. Both child nodes, corresponding to two

sub-clusters of the whole dataset, are themselves the root of a sub-tree representing the hierarchical bipartitions. More specifically, given as input:

- set X of the points to be clustered,
- the minimum number *min-region* of points for each cluster,
- the threshold value *split-value*, which refers to the components of the eigenvector associated to the smallest non-zero eigenvalue of the Laplacian matrix, and
- the value *min-region* of the *ncut* objective function, above which it is still possible to split the data,

the core of the algorithm can be summarized as follows:

1. if the current number of elements in X is lower than *min-region* **and** the corresponding *ncut* value lower that *min-region*, then stop the return directly X and stop the recursion;
2. if the stop-condition is not satisfied, then build the similarity matrix of all the points in X and the corresponding *degree* and normalized *Laplacian* matrices;
3. compute the first non zero eigenvalue λ of the normalized Laplacian and the corresponding eigenvector ϕ;
4. compute the by-partition of X by comparing the components of ϕ against the threshold *split-value*, and call X_1 and X_2 the resulting subgroups;
5. apply recursively the algorithm to both X_1 and X_2.

A concrete example of clusters generated by the implemented algorithm is shown in Figure 4.8.

4.3.2 Similarity measures for the blobs

It is well assessed that, in the spectral approach, clustering results strongly depend on the ability to capture the underlying notion of metric - or the similarity structure - over the input space. In turn, the similarity structure is related to the choice of a proper kernel function on the data. A function to be used as a kernel for spectral clustering should be symmetric and positive. Since, in our experiments we compare different representation schemes, we have to choose the corresponding kernel functions appropriately. In all the experiments presented in this paper we used a Gaussian kernel for texture and position and the histogram intersection kernel for color features (see [OBV05] for insightful discussion on the use of appropriate kernels

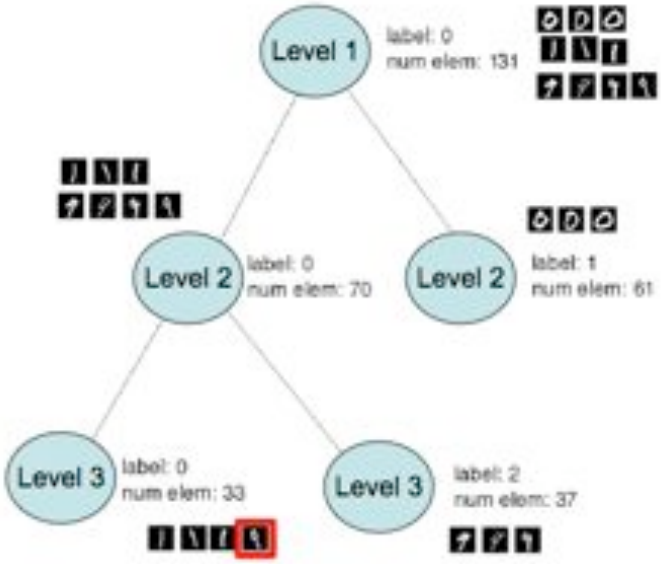


Figure 4.8: Graphical representation of the recursive clustering of a sub-sampling of the whole *MNIST Handwritten Digits* dataset comprising only the digits 0, 1, and 9.

for images). More formally, we define X_i the feature vector composed by position, color and texture features as follow

$$X_i = \{p_{i1}, p_{i2}, t_{i1}, \ldots, t_{in_t}, c_{i1}, \ldots, c_{in_c}\}$$

with

- $Pos_i = \{p_{i1}, p_{i2}\}$ as relative position of image blob,
- $Tex_i = \{Tex_{i\mu}, Tex_{i\epsilon}\} = \{(\mu_1, \ldots, \mu_{n_t/2}), (\epsilon_1, \ldots, \epsilon_{n_t/2})\}$ as means and variances of the filter responses discussed in Section 4.2,
- $Col_i = \{c_{i1}, \ldots, c_{in_c}\}$ as HS color histogram, and
- n_t and n_c are respectively the size of color feature vector and texture feature vector.

Therefore, we build up the compound kernel by taking the convex combination of the kernels functions for each cue:

$$K_{TOT} = \alpha K_{pos} + \beta K_{tex} + \gamma K_{col},$$

where

$$K_{pos} = e^{\frac{-||Pos_i - Pos_j||^2}{\sigma^2_{pos}}},$$

$$K_{text} = e^{\frac{-||Tex_{i\mu} - Tex_{j\mu}||^2}{\sigma_\mu^2}} e^{\frac{-||Tex_{i\epsilon} - Tex_{j\epsilon}||^2}{\sigma_\epsilon^2}}$$

and

$$K_{col} = \sum_{l=1}^{n_c} \min(Col_{il}, Col_{jl}).$$

4.4 Automatic labeling

Once we have collected a number of clusters the final step is to assign a semantic label to each blob in (possibly) all the clusters. As anticipated we want to approach this problem in an unsupervised way, that is to say without asking a human user to decide the labels of all the blobs that represent a specific concept. However, we are aware that this task would be quite impossible without the use of some form of prior knowledge because otherwise we could be easily misled by some geometrical property of the collected dataset which are irrelevant from a semantic viewpoint. Since a frequent form of knowledge about generic categories of the images are available in the form of multiple tags assigned to the images, we decided to use such weak form of prior knowledge. Recall that tags refers to images as a whole while our approach is based on image parts. The resulting scenario is schematically showed in Figure **??**.

We assume there exists a training collection T, of tagged images. On the one side, each image $\mathbf{I}_i \in T$ is connected to one or more tags $L_1^i, \ldots, L_k^i$. On the other side, the same image is connected to one or more clusters of blobs $C_1^i, \ldots, C_h^i$. Our proposal is to use the histograms of tags and their variations in order to find out possible direct connections between a cluster and a label. Specifically, for each of the clusters C_i^j, with $j = 1, \ldots, h$ we aim at finding out which tag L_i^* is the most probable candidate to represent the visual content of the blobs contained in C_i^j. If such probabilistic association is significant, then we use the L_i^* as the name of the semantic concept ℓ_i^j and proceed further by building the supervised classifier that represents algorithmically the concept. The basic idea underlying the approach can be explained better by means of some concrete examples. As discussed in the next section, we used the COREL30K database in which each image is associated with 4 or 5 generic tags like, e.g., *sky*, *people*, *trees*, *mountains*, *leaves*, *snow*, *grass* and similar. The histogram of the most frequent tags is shown in Figure 4.10.

In order to limit the variability of the visual pattern present in the set of images and obtain more consistent results, we sub-sample the whole data set by selecting all the images in the database that are associated to a specific concept. For examples, if we want to consider images that are likely to represent outdoor scenes, we extract all the images labeled with the tag *trees*. This approach is a simple way to cope with the huge number of data and to restrict the enormous variability of blobs appearance. Of course if one builds the histogram of the tags relative to the blobs extracted from this sub-sample of images the result is different from

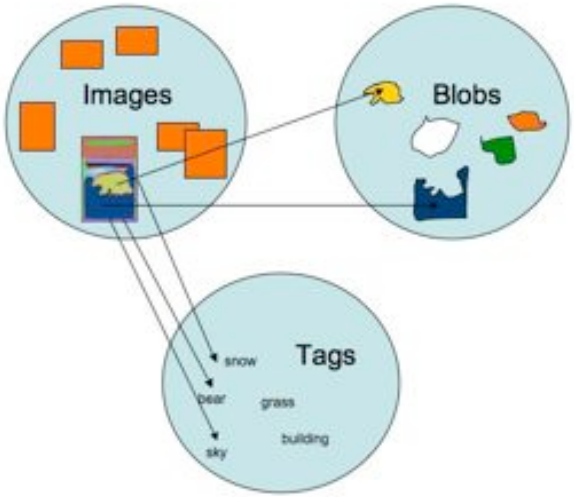


Figure 4.9: Schematic representation of the relationships between images, their tags, and the set of all the image parts obtained from the segmentation module. The images are (coarsely) tagged as a whole, i.e. the tags are assigned by looking both foreground objects and their context. However our approach requires (almost) exact labeling of the blobs.

that relative to the whole dataset. We adopt a simple approach to assign a tag to all the blobs in the same cluster, based on differences between histograms of tag occurrence before and after the clustering step. For each cluster of blobs, the starting point for the subsequent analysis is the comparison between the histogram shown in Figure 4.11(a) and the histograms relative to the distribution of tags for each cluster.

Figure 4.11(b) represents the histogram of the differences of the occurrence frequency for a fixed tag. Indeed, such flat pattern is the most likely in our experiments and we cannot infer any substantially new information. However, there are some clusters for which the level of *novelty* is higher, that is to say there are histograms of differences whose pattern present a pronounced peak in correspondence of a specific tag. A clear example is shown in Figure 4.11(c). In our approach we select all the clusters whose histogram presents such peak and define tentatively the tag associated to the peak as the semantic concept associated to the visual pattern represented by the blobs in the cluster. Figure 6.1 shows some *categorized* blobs in a number of images, showing that the propose approach is able to find out correctly visual patterns associated to some specific concepts.

In the previous examples, it is not possible to extract the semantic concept *trees* because we used this tag for the initial query and necessarily the tag *trees* is present the images associated

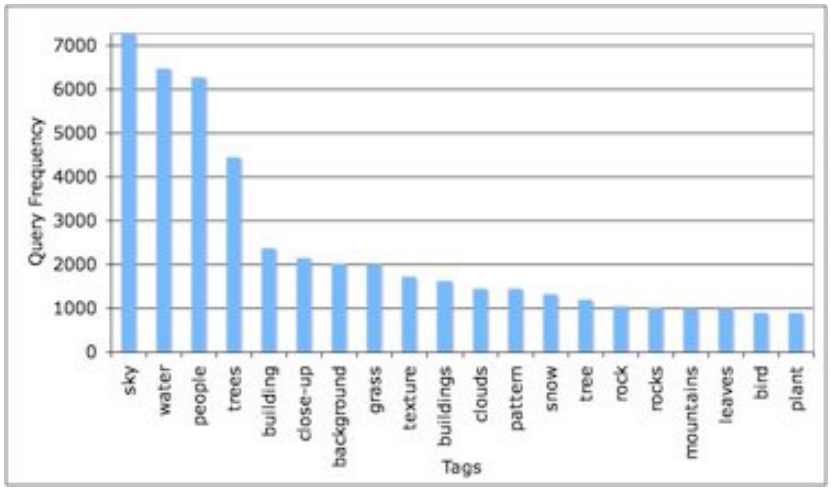


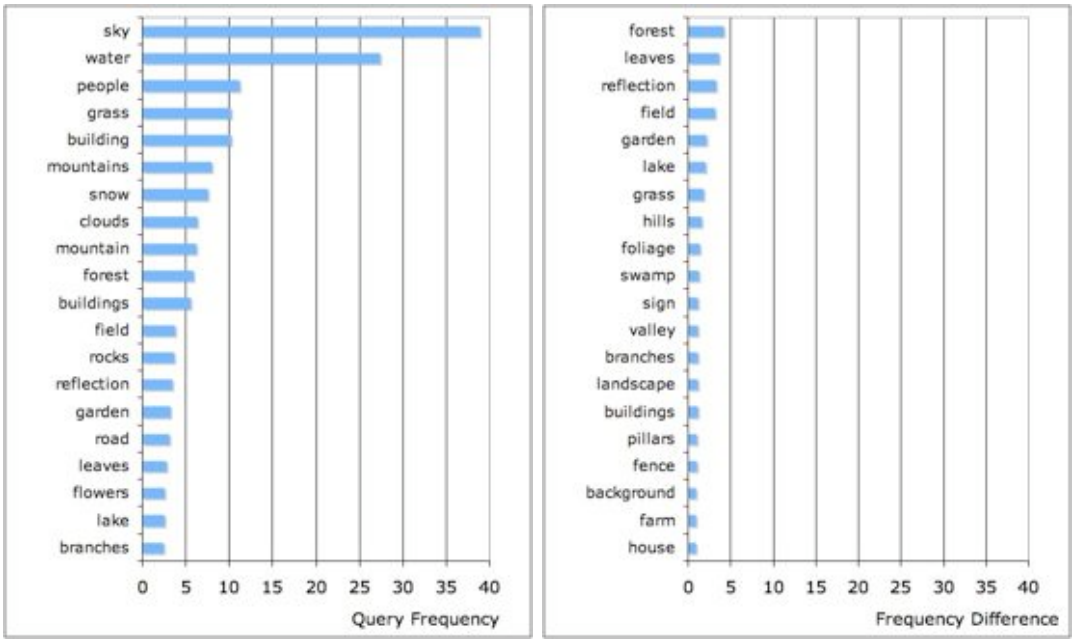
Figure 4.10: Histograms of the occurrence of the most frequent tags in the COREL30K dataset.

to all the cluster. In order to extract such concept, one needs to choose a different tag for the query. For example in Figure 4.11(d) we show the histogram of the candidate cluster for *trees* which has been obtained with the query mountains. Interestingly, the grouping algorithm yield two or more clusters where the histograms of the differences present a peaks for the same concept. This means that in those specific cases our approach is able to extract sub-concepts, for example cloudy versus clear skies or trees seen at different scales.

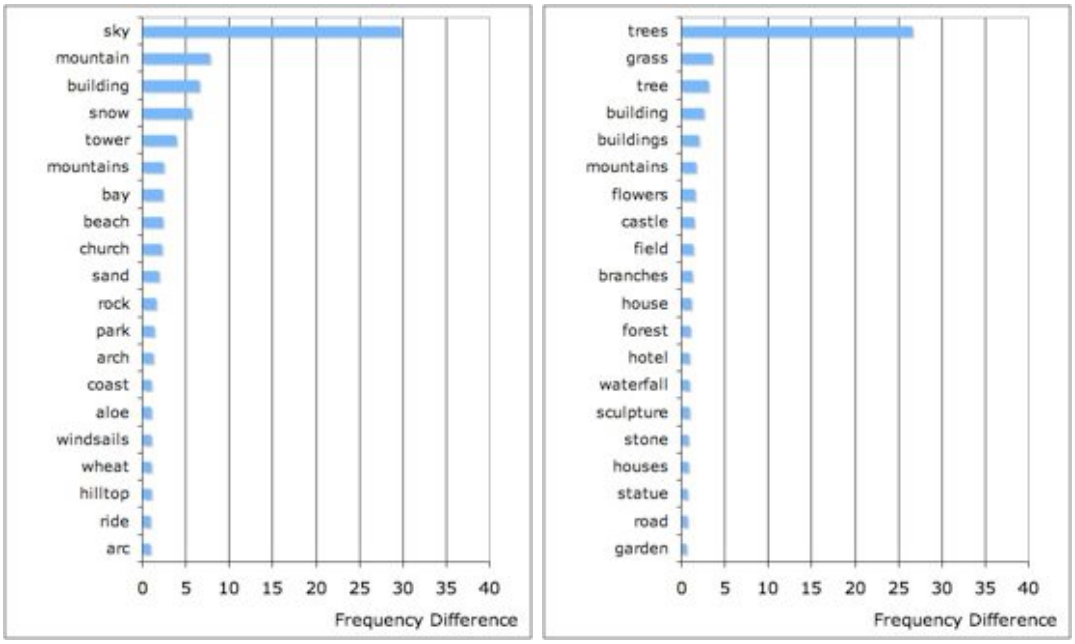
4.5 Dataset issues and discussion

In this section we present some issues concerning databases in annotation and retrieval context, focusing our analysis on the dataset exploited for an experimental evaluation of our system, the COREL30K dataset.

In the repertoire of images under consideration there is a gradual distinction between narrow and broad domains (see [SWS+00]). In a *narrow domain*, one finds a limited variability of the image content. An example of a narrow domain is a set of frontal views of faces recorded against a clear background. Although each face is unique and has large variability in the visual details, there are obvious geometrical, physical, and color-related constraints governing the domain. A *broad domain* has an unlimited and unpredictable variability in its appearance even for the same semantic meaning. In broad domains, images are polysemic and their semantics are described only partially. It might be the case that there are conspicuous objects in the scene for which the object class is unknown or even that the interpretation of the scene is not unique.



(a) Subsample (b) No Association



(c) Sky (d) Trees

Figure 4.11: An example of the frequency differences from which we derive the potential concept associated to each cluster of blobs. Figure 4.11(a), analogously to the Figure 4.10, shows the most frequent tags in a sub-sample of the Corel dataset. Figures 4.11(b), 4.11(c) and 4.11(d) show the bin-to-bin differences between the histogram in Figure 4.11(a) and the histogram of tags for each cluster.

In recent years, annotation tools provide large annotation databases by relying on the collaborative effort of users. Ponce et al. [PBE+06] discuss on the lacks about some datasets, exploited for evaluating the performance of object recognition and scene categorization tasks and suggest some criteria for gathering future datasets. The key issue is that algorithms, theoretically conceived for broad domains, are experimentally evaluated on a narrow domain. In this sense, COREL30K dataset represents a excellent repository of images, in which labels are assigned to whole images. Nevertheless, COREL30K has several drawbacks related to image labeling. The first issue concerns a language problem: a partial solution is achieved by using a toolbox for identifying a list of possible synonyms for each tag [PBE+06]. Second issue is due to the subjectivity of judgement: a number of images belonging to COREL30K dataset does not contain keywords representative of clearly visible objects and viceversa.

In this chapter we showed a way to build a reliable training set of representative image parts in order to use supervised methods for annotation and retrieval. In our approach we employ an articulated preprocessing stage in which we first compute image blobs, which are represented by means of color, texture and position based features, and then cluster them in order to discover meaningful subgroups that can be likely associated to semantic concepts.
As mentioned in Chapter 1, in Chapter 5 we will show the supervised approach applied to the training sets build automatically in this phase. Finally, in Chapter 6 we will see from the experimental viewpoint the problems on COREL30K dataset described above.

Chapter 5

Spectral learning with application to image annotation

The major advantage of the previous stage of unsupervised categorization and labeling of the blobs is that we can create automatically a training set of positive and negative examples of the visual appearance of a specific concept. At this point of the analysis, it is possible to include in the algorithmic architecture for annotation a number of supervised learning classifiers that learns from the data to categorize all the blobs extracted from the images in the database for the subsequent annotation stage.

In this chapter we describe the algorithmic solution proposed as learning machines, which is based on spectral regularization [LGRO$^+$08]. The proposed method has many interesting features: from the algorithmic point of view it is simple to implement, amounting to a few lines of code in Matlab. Moreover, it is appealing for many applications: the model selection is simple since it depends on few parameters, and over-fitting may be dealt in a very transparent way. Furthermore, the algorithm is faster than regularized least squares-based methods, without compromising classification performance.

In Section 5.1 we first discuss on the relationships between regularization and filtering in the context of inverse problems. In Section 5.2 we discuss previous work on filtering in the context of signal processing and learning. Section 5.3 reviews the regularized least-squares algorithm 3.2.5 from a filter function perspective. Section 5.4 is devoted to extending the filter point of view to a large class of kernel methods. In Section 5.5 we give several examples of the algorithms presenting two methods, such as the ν-method and iterated Tikhonov, that are new to learning. In Section 5.6 we discuss the properties of such algorithms and their complexity and Section 5.7 is left to a final discussion on the specific use of the algorithm within our general schema for image annotation and retrieval.

5.1 Relationships between regularization and filtering

As discussed in [BB98] one can regard regularization from a signal processing perspective introducing the notion of *filter*. This point of view gives a way to look constructively at regularization, where each regularization operator can be defined using spectral calculus as a suitable filter on the eigendecomposition of the operator defining the problem. The filter is designed to suppress the oscillatory behavior corresponding to small eigenvalues. In this view it is known, for example, that Tikhonov regularization can be related to Wiener filter [BB98].

As we mentioned, regularization has a long history in learning and our starting point is the theoretical analysis proposed in the papers [BPR06], [DVRV05], [CDV06], and [Cap06] showing that many regularization methods originally proposed in the context of inverse problems give rise to consistent kernel methods with optimal minimax learning rates. The analysis we propose in this chapter focuses on two points.

First, differently from [BPR06], we propose a more intuitive derivation of regularization based on the notion of spectral filter. We start introducing the notion of filter functions and explain why, besides ensuring numerical stability, they can also provide a way to learn with generalization guarantees. This requires, in particular, to discuss the interplay between filtering and random sampling. Our analysis is complementary to the theory developed in [BPR06], [DVRV05], [YRC07], [Cap06]. Note that the fact that algorithms ensuring numerical stability can also learn is not obvious but confirms the deep connection between stability and generalization (see [BE02], [PRMN04] and [RMP05] for references).

Second, we present and discuss several example of filters inducing spectral algorithms for supervised learning. The filter function perspective provides a unifying framework for discussing similarities and differences between the various methods. Some of these algorithms, such as the ν-method and iterated Tikhonov, are new to learning. Other algorithms are well known: spectral cut-off (TSVD) is related to Principal Component Regression (PCR) and its kernel version; Landweber iteration is known as L2-boosting [BY02] and Tikhonov regularization is also known as regularized least squares or ridge regression. Our analysis highlights the common regularization principle underlying algorithms originally motivated by seemingly unrelated ideas: *penalized empirical risk minimization* – like the regularized least-squares, *early stopping of iterative procedures* – like the gradient descent, and *(kernel) dimensionality reduction methods* – like (kernel) Principal Component Analysis.
Despite these similarities, spectral algorithms have differences from both the computational and theoretical point of view. One of the main differences is related to the so called *saturation* effect affecting some regularization schemes. This phenomenon, which is well known in inverse problems theory, amounts to the impossibility, for some algorithms, to exploit the regularity of the target function beyond a certain critical value, referred to as the *qualification* of the method. We try to shed light on this aspect, which is usually not discussed in literature of learning rates, via some theoretical considerations and numerical simulations.

Another crucial point that differentiates the spectral algorithms we study concerns algorithmic complexity. Some of the algorithms exhibit interesting computational properties: a relevant aspect is the built-in property of iterative methods to recover solutions corresponding to the whole regularization path [HTZ04]. We now move on describing the filtering perspective to learning (or to regularization).

5.2 Spectral filtering

We start with an overview of filtering approaches proposed in the field of machine learning. The notion of filter function was previously studied in machine learning and provides a connection to the literature of function approximation in signal processing and approximation theory. The pioneering work of [PG92] established the relation between neural networks, radial basis functions and regularization. This paper, as well as [GJP95] are suitable sources for references and discussions.

An important aspect that we would like to stress is that, from a technical point of view, these works (implicitly) assume the data to be sampled according to a uniform distribution and make an extensive use of Fourier theory. The extension to general probability distribution is not straightforward and this is crucial since it is standard in learning theory to assume the point to be drawn according to a *general, unknown* distribution. A mathematical connection between sampling theory and learning theory has been recently proposed in [SZ04, SZ05b] whereas [DVRC$^+$05, DVRC06] gave an inverse problem perspective to learning. The analysis we present can be seen as a further step towards a deeper understanding of learning as a function approximation problem.

Recently, filtering of the kernel matrix have been considered in the context of graph regularization – see for example [HTF01], [CWS03], [ZKG$^+$05], [SK03] and [ZA06] and references therein. In this case, reweighing (filtering) of a kernel matrix on a set of labeled and unlabeled input points is used to define new penalty terms replacing the square of the norm in the adopted hypothesis space. It has been shown – see for example [ZA06] – that this is equivalent to standard regularized least square with data dependent kernels. Note that in graph regularization no sampling is considered and the problem is truly a problem of transductive learning (i.e. considering a set of test points of interest – see for example [Vap98]).

Our analysis relies on a different use of filter functions to define *new algorithms* rather than *new kernels*. In fact in our setting the kernel is fixed and each rescaling of the kernel matrix leads to a learning algorithm which is not necessarily a penalized minimization. From the theoretical standpoint the dependency of the rescaling on the regularization parameter allows us to derive consistency results in a natural way. We now explore the well known Regularized Least-Squares from a filtering perspective and then move on analyzing other spectral algorithms.

5.3 Regularized Least-Squares as a spectral filter

In this section we review how the generalization property of the regularized least-squares algorithm is a consequence of the algorithm being seen as a filter on the eigenvalues of the kernel matrix. This point of view naturally suggests a new class of learning algorithms defined in terms of filter functions, the properties of which are discussed in the next section.

As mentioned in Chapter 3 in the framework of supervised learning, the regularized least-squares algorithm is based on the choice of a Mercer kernel[1] $K(\mathbf{x}, \mathbf{t})$ on the input space X and of a regularization parameter $\lambda > 0$. Hence, for each training set $\mathbf{z} = (\mathbf{x}, \mathbf{y}) = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$ of n-examples $(\mathbf{x}_i, y_i) \in X \times \mathbb{R}$, regularized least squares can be written as

$$f_{\mathbf{z}}^{\lambda}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i) \quad \text{with} \quad \alpha = (\mathbf{K} + n\lambda I)^{-1}\mathbf{y}. \tag{5.1}$$

where $\mathbf{K}$ is the $n \times n$-matrix $(\mathbf{K})_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$.

We start re-writing the equation (5.1) in a slightly different way

$$(f_{\mathbf{z}}^{\lambda}(\mathbf{x}_1), \ldots, f_{\mathbf{z}}^{\lambda}(\mathbf{x}_n)) = \frac{\mathbf{K}}{n}(\frac{\mathbf{K}}{n} + \lambda)^{-1}\mathbf{y}. \tag{5.2}$$

Observe that, if v is an eigenvector of $\mathbf{K}/n$ with eigenvalue σ, then we have $\frac{\mathbf{K}}{n}(\frac{\mathbf{K}}{n} + \lambda)^{-1}v = \frac{\sigma}{\sigma+\lambda}v$, so that the regularized least-squares algorithm is in fact a filter on the eigenvalues of the kernel matrix.
The filter $\frac{\sigma}{\sigma+\lambda}$ not only ensures numerical stability, but also the generalization properties of the estimator. To obtain a deep insight on this point, consider the population case when we have knowledge of the probability distribution ρ generating the data. In this setting, the kernel matrix $\mathbf{K}/n$ is replaced by the integral operator L_K with kernel K

$$L_K f(\mathbf{x}) = \int_X K(\mathbf{x}, \mathbf{s})\, f(\mathbf{s}) d\rho_X(\mathbf{s}) \qquad f \in L^2(X), \tag{5.3}$$

the data $\mathbf{y}$ is replaced by the regression function f_ρ, so that (5.2) becomes

$$f^{\lambda} = L_K(L_K + \lambda I)^{-1} f_\rho. \tag{5.4}$$

More explicitly, since L_K is a positive compact operator bounded by 1 and $\mathcal{H}$ is dense in $L^2(X)$, there is basis $(u_i)_{i\geq 1}$ in $L^2(X)$ such that $L_K u_i = \sigma_i u_i$ with $0 < \sigma_i \leq 1$ and $\lim_{i\to\infty} \sigma_i = 0$. Hence

$$\begin{aligned} f_\rho &= \sum_{i=1}^{\infty} \langle f_\rho, u_i \rangle_\rho \, u_i \\ f^{\lambda} &= \sum_{i=1}^{\infty} \frac{\sigma_i}{\sigma_i + \lambda} \langle f_\rho, u_i \rangle_\rho \, u_i. \end{aligned}$$

[1] This means that $K : X \times X \to \mathbb{R}$ is a symmetric continuous function, which is positive definite [Aro50].

By comparing the two equations, one has that f^λ is a good approximation of f_ρ, provided that λ is small enough. For such λ, the filter $\frac{\sigma}{\sigma+\lambda}$ selects only the components of the f_ρ corresponding to large eigenvalues, which are a finite number since the sequence of eigenvalues goes to zero. Hence, if we slightly perturb both L_K and f_ρ, the corresponding solution of (5.3) is close to f_ρ, provided that the perturbation is small. The key idea is that now we can regard the sample case $\mathbf{K}$, $\mathbf{y}$ and the corresponding estimator $f^\lambda_{\mathbf{z}}$, as perturbation of L_K, f_ρ and f^λ, respectively. For a mathematical proof of the above intuition we refer the reader to [DVRV05, BPR06].

The above discussion suggests that one can replace $\frac{\sigma}{\sigma+\lambda}$ with other functions $\sigma g_\lambda(\sigma)$ which are filters on the eigenvalues of L_K and obtain different regularization algorithms, as shown in the next section.

5.4 Properties of spectral filters

In this section we discuss the properties of kernel methods based on spectral filtering. Our approach is inspired by inverse problems. A complete theoretical discussion of our approach can be found in [DVRV05, BPR06, Cap06].

Starting from (5.1), the conclusions drawn in the previous section suggest to define a new class of learning algorithm by letting

$$f^\lambda_{\mathbf{z}} = \sum_{i=1}^n \alpha_i K(\mathbf{x}, \mathbf{x}_i) \qquad \text{with} \qquad \alpha = \frac{1}{n} g_\lambda(\frac{\mathbf{K}}{n})\mathbf{y}, \tag{5.5}$$

Here $g_\lambda(\frac{\mathbf{K}}{n})$ is defined by spectral calculus, that is, if v is an eigenvector of $\mathbf{K}/n$ with eigenvalue σ, then $g_\lambda(\frac{\mathbf{K}}{n})v = g_\lambda(\sigma)v$. In particular, on the given data, one has

$$(f^\lambda_{\mathbf{z}}(\mathbf{x}_1), \ldots, f^\lambda_{\mathbf{z}}(\mathbf{x}_n)) = \frac{\mathbf{K}}{n}\ g_\lambda(\frac{\mathbf{K}}{n})\mathbf{y}. \tag{5.6}$$

We note that, unlike regularized least squares, such an estimator is not necessarily the solution of penalized empirical minimization. Clearly, to ensure both numerical stability as well as consistency, we need to make some assumptions on g_λ. Following [DVRV05, BPR06] we say that a function g_λ is an admissible filter function if:

1. There exists a constant B such that
$$\sup_{0<\sigma\leq 1} |g_\lambda(\sigma)| \leq \frac{B}{\lambda} \qquad \forall \lambda \in [0,1]. \tag{5.7}$$

2. There exists a constant D such that
$$\begin{aligned} \lim_{\lambda\to 0} \sigma g_\lambda(\sigma) &= 1 \quad \forall \sigma \in\ ]0,1] \\ \sup_{0<\sigma\leq 1} |\sigma g_\lambda(\sigma)| &\leq D \quad \forall \lambda \in [0,1]. \end{aligned} \tag{5.8}$$

3. There is a constant $\overline{\nu} > 0$, namely *the qualification of the regularization* g_λ such that

$$\sup_{0<\sigma\leq 1} |1 - g_\lambda(\sigma)\sigma|\sigma^\nu \leq \gamma_\nu \lambda^\nu, \qquad \forall\ 0 < \nu \leq \overline{\nu}, \tag{5.9}$$

   where the constant $\gamma_\nu > 0$ does not depend on λ.

A simple computation shows that $g_\lambda(\sigma) = \frac{1}{\sigma+\lambda}$ is an admissible filter function, indeed Eqs. (5.7) and (5.8) hold with $B = D = 1$, condition (5.9) is verified with $\gamma_\nu = 1$ for $0 < \nu \leq 1$ and hence the qualification equals to 1. Other examples are discussed in the next section. Here we give a heuristic motivation of the above conditions having in mind the discussion in the previous section. First, observe that population version of (5.5) becomes

$$f^\lambda = \sum_i \sigma_i g_\lambda(\sigma_i) \langle f_\rho, u_i \rangle_\rho u_i. \tag{5.10}$$

We can make the following observations.

1. Eq. (5.7) ensures that eigenvalues of $g_\lambda(\mathbf{K})$ are bounded by $\frac{B}{\lambda}$, so that (5.5) is numerically stable. Moreover, looking at (5.10), we see that it also implies that, if σ_i is much smaller than λ, the corresponding Fourier coefficient $\langle f^\lambda, u_i \rangle$ of f^λ is small. Hence, f^λ has essentially only a finite number of non-zero Fourier coefficients on the basis $(u_i)_{i\geq 1}$ and we can argue that, by the law of large numbers, $f^\lambda_{\mathbf{z}}$ is a good approximation of f^λ when n is large enough.

2. Assumption (5.8) implies that f^λ converges to f_ρ if λ goes to zero. In terms of the kernel matrix, such a condition means that $g_\lambda(\mathbf{K})$ converges to $\mathbf{K}^{-1}$ when λ goes to zero, avoiding over-smoothing.

3. Condition (5.9) is related to the convergence rates of the algorithm. These rates depend on how fast the Fourier coefficients $\langle f_\rho, u_i \rangle$ converge to 0 with respect to the eigenvalues σ_i [BPR06]. This information is encoded by a priori assumptions on f_ρ of the form

$$\sum_{i=1}^{\infty} \frac{\langle f_\rho, u_i \rangle_\rho^2}{\sigma_i^{2r}} < R, \tag{5.11}$$

   where the parameter r encodes the regularity property of the regression function. If $r = 1/2$ this corresponds to assuming $f_\rho \in \mathcal{H}$ and, more generally, the larger is r the smoother is the function. Condition (5.9) and the choice $\lambda_n = \frac{1}{n^{2r+1}}$ ensure that, if $r \leq \overline{\nu}$,

$$\left\| f^{\lambda_n}_{\mathbf{z}} - f_\rho \right\|_\rho \leq C n^{-\frac{r}{2r+1}} \quad \text{with high probability,} \tag{5.12}$$

   whereas, if $r \geq \overline{\nu}$, the rate of convergence is always $n^{-\frac{\overline{\nu}}{2\overline{\nu}+1}}$; for a proof and a complete discussion see [BPR06, Cap06].

Hence filter functions having a larger qualification $\overline{\nu}$ gives better rates, that is, the corresponding algorithms can better exploit the smoothness of f_ρ. This fact marks a big distinction among the various algorithms we consider as we discussed in the following.

Considering the decomposition $f_{\mathbf{z}}^\lambda - f_\rho = (f_{\mathbf{z}}^\lambda - f^\lambda) + (f^\lambda - f_\rho)$, from the above discussion we have that the consistency of this class of learning algorithms depends on two opposite terms: the approximation error $\left\|f^\lambda - f_\rho\right\|$ and the sample error $\left\|f_{\mathbf{z}}^\lambda - f^\lambda\right\|$. The approximation error depends on the examples only through $\lambda = \lambda_n$ and it decreases if λ goes to zero, whereas the sample error is of probabilistic nature and it increases if λ goes to zero. The optimal choice of the regularization parameter λ will be a trade-off between these two errors – see [DVCR05, CDV06, SZ05a, WYZ06] and references therein about the rates for regularized least-squares, and [DVRV05, BPR06, Cap06] for arbitrary filters.

Before giving several examples of algorithms fitting into the above general framework we observe that the considered algorithms can be regarded as filters on the expansion of the target function on a suitable basis. In principle, this basis can be obtained from the spectral decomposition of the integral operator L_K and, in practice, is approximated by considering the spectral decomposition of the kernel matrix $\mathbf{K}$. Interestingly the basis thus obtained has a natural interpretation: if the data are centered (in the feature space), then the elements of the basis are the principal components of the expected (and empirical) covariance matrix in the feature space. In this respect the spectral methods we discussed rely on the assumption that most of the information is actually encoded in the first principal components.

5.5 Filter algorithms

In this section we give some specific examples of kernel methods based on spectral regularization. All these algorithms are known in the context of regularization for linear inverse problems but only some of them have been used for statistical inference problems. These methods have many interesting features: from the algorithmic point of view they are simple to implement, usually they amount to a few lines of code. They are appealing for applications: their model selection is simple since they depend on few parameters, while over-fitting may be dealt in a very transparent way. Some of them represent a very good alternative to Regularized Least Squares as they are faster without compromising classification performance (see Section 6.1.1). Note that for regularized least squares the algorithm has the following variational formulation

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2 + \lambda \left\|f\right\|_{\mathcal{H}}^2$$

which can be interpreted as an extension of empirical risk minimization. In general the class of regularization might not be described by a variational problem so that filter point of view provides us with a suitable description.

More details on the derivation of these algorithms can be found in [EHN96].

5.5.1 Iterative Landweber

Landweber iteration is characterized by the filter function

$$g_t(\sigma) = \tau \sum_{i=0}^{t-1} (1 - \tau\sigma)^i$$

where we identify $\lambda = t^{-1}, t \in \mathbb{N}$ and take $\tau = 1$ (since the kernel is bounded by 1). In this case we have $B = D = 1$ and the qualification is infinite since (5.9) holds with $\gamma_\nu = 1$ if $0 < \nu \leq 1$ and $\gamma_\nu = \nu^\nu$ otherwise. The above filter can be derived from a variational point of view. In fact, as shown in [YRC07], this method corresponds to empirical risk minimization via gradient descent. If we denote with $\|\cdot\|_n$ the norm in $\mathbb{R}^n$, we can impose

$$\nabla \|\mathbf{K}\alpha - \mathbf{y}\|_n^2 = 0,$$

and by a simple calculation we see that the solution can be rewritten as the following iterative map

$$\alpha_i = \alpha_{i-1} + \frac{\tau}{n}(\mathbf{y} - \mathbf{K}\alpha_{i-1}), \qquad i = 1, \ldots, t$$

where τ determines the step-size. We may start from a very simple solution, $\alpha_0 = 0$. Clearly if we let the number of iterations grow we are simply minimizing the empirical risk and are bound to overfit. Early stopping of the iterative procedure allows us to avoid over-fitting, thus the iteration number plays the role of the regularization parameter. In [YRC07] the fixed step-size $\tau = 1$ was shown to be the best choice among the variable step-size $\tau = \frac{1}{(t+1)^\theta}$, with $\theta \in [0, 1)$. This suggests that τ does not play any role for regularization. Landweber regularization was introduced under the name of L2-boosting for splines in a fixed design statistical model [BY02] and eventually generalized to general RKH spaces and random design in [YRC07].

5.5.2 Semi-iterative Regularization

An interesting class of algorithms are the so called semi-iterative regularization or accelerated Landweber iteration. These methods can be seen as a generalization of Landweber iteration where the regularization is now

$$g_t(\sigma) = p_t(\sigma)$$

with p_t a polynomial of degree $t-1$. In this case we can identify $\lambda = t^{-2}, t \in \mathbb{N}$. One can show that $D = 1$, $B = 2$ and the qualification of this class of methods is usually finite [EHN96].

An example which turns out to be particularly interesting is the so called $\nu - method$. The derivation of this method is fairly complicated and relies on the use of orthogonal polynomials

to obtain acceleration of the standard gradient descent algorithm (see chapter 10 in [GVL96]). Such a derivation is beyond the scope of this presentation and we refer the interested reader to [EHN96]. In the $\nu - method$ the qualification is ν (fixed) with $\gamma_\nu = c$ for some positive constant c. The algorithm amounts to solving (with $\alpha_0 = 0$) the following map

$$\alpha_i = \alpha_{i-1} + u_i(\alpha_{i-1} - \alpha_{i-2}) + \frac{\omega_i}{n}(\mathbf{y} - \mathbf{K}\alpha_{i-1}), \quad i = 1, \dots, t$$

where

$$\begin{aligned} u_i &= \frac{(i-1)(2i-3)(2i+2\nu-1)}{(i+2\nu-1)(2i+4\nu-1)(2i+2\nu-3)} \\ \omega_i &= 4\frac{(2i+2\nu-1)(i+\nu-1)}{(i+2\nu-1)(2i+4\nu-1)} \qquad t > 1. \end{aligned}$$

The interest of this method lies in the fact that since the regularization parameter here is $\lambda = t^{-2}$, we just need the square root of the number of iterations needed by Landweber iteration. In inverse problems this method is known to be extremely fast and is often used as a valid alternative to conjugate gradient – see Chapter 6 for details. To our knowledge semi-iterative regularization has not been previously in learning.

5.5.3 Spectral Cut-Off

This method, also known as truncated singular values decomposition (TSVD), is equivalent to the so called (kernel) principal component regression. The filter function is simply

$$g_\lambda(\sigma) = \begin{cases} \frac{1}{\sigma} & \sigma \geq \lambda \\ 0 & \sigma < \lambda \end{cases}$$

In this case, $B = D = 1$. The qualification of the method is arbitrary and $\gamma_\nu = 1$ for any $\nu > 0$. The corresponding algorithm is based on the following simple idea. Perform SVD of the kernel matrix $\mathbf{K} = \mathbf{U}\mathbf{S}\mathbf{U}^T$ where $\mathbf{U}$ is an orthogonal matrix and $\mathbf{S} = diag(\sigma_1, \dots, \sigma_n)$ is diagonal with $\sigma_i \geq \sigma_{i+1}$. Then discard the singular values smaller than the threshold λ, replace them with 0. The algorithm is then given by

$$\alpha = \mathbf{K}_\lambda^{-1}\mathbf{y} \tag{5.13}$$

where $\mathbf{K}_\lambda^{-1} = \mathbf{U}^T\mathbf{S}_\lambda^{-1}\mathbf{U}$ and $\mathbf{S}_\lambda^{-1} = diag(1/\sigma_1, \dots, 1/\sigma_m, 0, \dots)$ where $\sigma_m \geq \lambda$ and $\sigma_{m+1} < \lambda$. The regularization parameter is the threshold λ or, equivalently, the number m of components that we keep.

Finally, notice that, if the data are centered in the feature space, then the columns of the matrix $\mathbf{U}$ are the principal components of the covariance matrix in the feature space and the spectral cut-off is a filter that discards the projection on the last principal components. The procedure is well known in literature as kernel principal component analysis – see for example [SS02].

5.5.4 Iterated Tikhonov

We conclude this section mentioning a method which is a mixture between Landweber iteration and Tikhonov regularization. Unlike Tikhonov regularization which has finite qualification and cannot exploit the regularity of the solution beyond a certain regularity level, iterated Tikhonov overcomes this problem by means of the following regularization

$$g_\lambda(\sigma) = \frac{(\sigma+\lambda)^\nu - \lambda^\nu}{\sigma(\sigma+\lambda)^\nu}, \quad \nu \in \mathbb{N}.$$

In this case we have $D = 1$ and $B = t$ and the qualification of the method is now ν with $\gamma_\nu = 1$ for all $0 < \nu \leq t$. The algorithm is described by the following iterative map

$$(\mathbf{K} + n\lambda I)\alpha_i = \mathbf{y} + n\lambda\alpha_{i-1} \qquad i = 1, \ldots, \nu$$

choosing $\alpha_0 = 0$. It is easy to see that for $\nu = 1$ we simply recover the standard Tikhonov regularization but as we let $\nu > 1$ we improve the qualification of the method with respect to standard Tikhonov. Moreover we note that by fixing λ we can think of the above algorithms as an iterative regularization with ν as the regularization parameter.

5.6 Algorithmic complexity and regularization path

In this section we will comment on the properties of spectral regularization algorithms in terms of algorithmic complexity.

Having in mind that each of the algorithms we discussed depends on at least one parameter[2] we are going to distinguish between: (1) the computational cost of each algorithm for one fixed parameter value and (2) the computational cost of each algorithm to find the solution corresponding to many parameter values. The first situation corresponds to the case when a correct value of the regularization parameter is given a priori or has been computed already. The complexity analysis in this case is fairly standard and we compute it in a worst case scenario, though for nicely structured kernel matrices (for example sparse or block structured) the complexity can be drastically reduced.

The second situation is more interesting in practice since one usually has to *find* a good parameter value, therefore the real computational cost includes the parameter selection procedure. Typically one computes solutions corresponding to different parameter values and then chooses the one minimizing some estimate of the generalization error, for example hold-out or leave-one-out estimates [HTF01]. This procedure is related to the concept of *regularization path* [HTZ04]. Roughly speaking the regularization path is the sequence of solutions, corresponding to different parameters, that we need to compute to select the best parameter

[2] In general, besides the regularization parameter, there might be some kernel parameter. In our discussion we assume the kernel (and its parameter) to be fixed.

estimate. Ideally one would like the cost of calculating the regularization path to be as close as possible to that of calculating the solution for a fixed parameter value. In general this is a strong requirement but, for example, SVM algorithm has a step-wise linear dependence on the regularization parameter [PV98] and this can be exploited to find efficiently the regularization path [HTZ04].

Given the above premises, analyzing spectral regularization algorithms we notice a substantial difference between iterative methods (*Landweber* and *ν-method*) and the others. At each iteration, iterative methods calculate a solution corresponding to t, which is both the iteration number and the regularization parameter (as mentioned above, equal to $1/\lambda$). In this view iterative methods have the built-in property of computing the *whole* regularization path. Landweber iteration at each step i performs a matrix-vector product between $\mathbf{K}$ and α_{i-1} so that at each iteration the complexity is $O(n^2)$. If we run t iteration the complexity is then $O(t*n^2)$. Similarly to Landweber iteration, the ν-method involves a matrix-vector product so that each iteration costs $O(n^2)$. However, as discussed in Section 5.5, the number of iteration required to obtain the same solution of Landweber iteration is the square root of the number of iterations needed by Landweber (see also Table 6.2). Such rate of convergence can be shown to be optimal among iterative schemes (see [EHN96]). In the case of *RLS* in general one needs to perform a matrix inversion for each parameter value that costs in the worst case $O(n^3)$. Similarly for spectral cut-off the cost is that of finding the singular value decomposition of the kernel matrix which is again $O(n^3)$. Finally we note that computing solution for different parameter values is in general very costly for a standard implementation of RLS, while for spectral cut-off one can perform only one singular value decomposition. This suggests the use of SVD decomposition also for solving RLS, in case a parameter tuning is needed.

5.7 Application of supervised learning for the annotation and retrieval of images

In this chapter we presented and discussed several spectral algorithms for supervised learning. Starting from the standard regularized least squares we showed that a number of methods from the inverse problems theory lead to consistent learning algorithms. Specifically, we identified the iterative algorithm known as ν-method as the most appropriate for our purposes. One of its main advantages is the simplicity and the possibility to choose easily the optimal parameters and select the learning model that best suits our requirement.

Once the previous stages have been completed, our annotation and retrieval system comprises a database $R = \{\mathbf{I}_1, \ldots, \mathbf{I}_n\}$ of images and a semantic vocabulary $L = \{\ell_1, \ldots, \ell_T\}$ of semantic *visual concepts* ℓ_i learnt directly from the data. All the images can be associated with one or more concepts. The interesting peculiarity of the definition of *concept* in our system is the existence of a classifier able to recognize the local visual appearance associated to that specific concept. Therefore, our search engine is based on *typical* visual appearance associated to a

concept rather than on the occurrence of a specific set of keywords in a text-based annotation.

Given a specific concept ℓ_i, the goal of semantic retrieval is to extract the images in the repository that are directly related to ℓ_i. Instead, the goal of automatic image annotation is to extract the set of semantic labels $(\ell_1, \ldots, \ell_K)$ that can be associated to a given image $\mathbf{I}$ so that ℓ_i describes well the visual content of $\mathbf{I}$. We can now outline the procedures used by our system to perform both retrieval and annotation. Users can submit a semantic query in two different ways. First, they are allowed to type the name of the concept and, if it belongs to the list of known concepts then the system simply displays all the images of which at least one blob is classified positive by the classifier associated to the concept. Of course more complex strategies may be adopted to attribute a positive label to an image, such as those based on the simultaneous presence of multiple blobs positive (or nearly positive) in the same image or on some form of voting schema. Second, users are also allowed to input an image and ask for all the images in the repository which share the same semantic content. In such case, we preprocess the image by segmenting it into several blobs and attributing to each blob a concept ℓ_i according to the result of the classifiers. If a concept ℓ_i is depicted in the images, then we look for all the images in the database containing ℓ_i. Finally, users can add a new unannotated image in the repository. All the new images are segmented and the candidate semantic concepts represented by the image are extracted and prompted to the user to verify that the automatic annotation process is correct.

Chapter 6

Experimental evaluation

The theoretical results and the statistical methods presented in the previous chapters of the thesis provided us with suitable tools to build an automatic system for image annotation and retrieval. By leveraging on these tools, we designed a modular algorithmic pipeline and learned to recognize the *visual cues* associated to a number of semantic concepts present in our evaluation dataset. The learning approach adopted in this part of the system was unsupervised and was based on hierarchical spectral clustering. Once a number of concepts had been characterized, we exploited the learned visual cues to build supervised classifiers able to recognize image parts representing those concepts. Both image annotation and semantic retrieval have been achieved quite naturally by starting from the classification of the most representative parts of the image.
The goal of this chapter is to present a number of experiments conducted to evaluate the effectiveness of our algorithms and methods and test the potentials of the proposed approach to image annotation and retrieval. In the following, we describe the results by starting from the experimental validation of the core learning modules first, and then moving to the analysis of the system as a whole.
In order to cope with the large collections of images required by our unsupervised statistical approach, a further critical issue we had to address was to make the computational part extremely efficient. Actually, since most of the modules of the systems works on thousands of images separately we adopted suitable implementation strategies which allowed us to deploy effectively the Grid resources present in our department.

The rest of the chapter is organized as follows. In Section 6.1 we report an experimental analysis of the spectral algorithms introduced in Chapter 5. We recall that this first experimental analysis constitutes an original contribution of our work and has been published in [LGRO+08]. The analysis is based on a set of well known benchmark data, and we compare the obtained results with the ones reported in the literature.
Hence, section 6.2 presents preliminary results of the annotation and retrieval architecture build upon the original modules described in Chapters 4 and 5. Our analysis starts by showing

a number of randomly picked examples of blobs belonging to the same cluster. Subsequently, we evaluate the performance of the concept-based classifiers trained using the ν-method. Finally, in section 6.3 we describe in some detail the algorithmic Grid-based infrastructure we developed to conduct the experiments.

6.1 Experimental analysis of spectral algorithms

This section reports experimental evidence of the effectiveness of the algorithms discussed in Section 5.5. We apply them to a number of classification problems, first considering a set of well known benchmark data and comparing the results we obtain with the ones reported in the literature; then we consider a more specific application, face detection, analyzing the results obtained with a spectral regularization algorithm and comparing them with SVM, which has been applied with success in the past by many authors. For these experiments we consider both a benchmark dataset available on the web and a set of data acquired by a video-monitoring system designed in our lab.

6.1.1 Experiments on benchmark datasets

In this section we analyze the classification performance of the regularization algorithms on various benchmark datasets. In particular we consider the IDA benchmark, containing one toy dataset (`banana` — see Table 6.1), and several real datasets[1]. These datasets have been previously used to assess many learning algorithms, including Adaboost, RBF networks, SVMs, and Kernel Projection Machines. The benchmarks webpage reports the results obtained with these methods and which for our comparisons.

For each dataset, 100 resamplings into training and test sets are available from the website. The structure of our experiments follows the one reported on the benchmarks webpage: we perform parameter estimation with 5-fold cross validation on the first 5 partitions of the dataset, then we compute the median of the 5 estimated parameters and use it as an optimal parameter for all the resamplings. As for the choice of *parameter* σ (i.e., the standard deviation of the RBF kernel), at first we set the value to the average of square distances of training set points of two different resamplings: let it be σ_c. Then we compute the error on two randomly chosen partitions on on the range $[\sigma_c - \delta, \sigma_c + \delta]$ for a small δ, on several values of λ and choose the most appropriate σ. After selecting σ, the *parameter* t (corresponding to $1/\lambda$) is tuned with 5-CV on the range $[1, \infty]$ where κ is $\sup_{x \in X} K(x, x)$. Regarding the choice of the parameter ν for the $\nu - method$ and iterated Tikhonov (where ν is the number of iteration) we tried different values obtaining very similar results. The saturation effect on real data

[1] This benchmark is available at the website:
`http://ida.first.fraunhofer.de/projects/bench/`.

Table 6.1: The 13 benchmark datasets used:their size (training and test), the space dimension and the number of splits in training/test.

	#Train	#Test	Dim	#Resampl.
(1)Banana	400	4900	2	100
(2)B.Canc.	200	77	9	100
(3)Diabet.	468	300	8	100
(4)F.Solar	666	400	9	100
(5)German	700	300	20	100
(6)Heart	170	100	13	100
(7)Image	1300	1010	18	20
(8)Ringn.	400	7000	20	100
(9)Splice	1000	2175	60	20
(10)Thyroid	140	75	5	100
(11)Titanic	150	2051	3	100
(12)Twonorm	400	7000	20	100
(13)Wavef.	400	4600	21	100

seemed much harder to spot and all the errors where very close. In the end we chose $\nu = 5$ for both methods.

Table 6.2 shows the average generalization performance (with standard deviation) over the data sets partitions. It also reports the parameters σ and t $(= 1/\lambda)$ chosen to find the best model. The results obtained with the five methods are very similar, with the exception of Landweber whose performances are less stable. The $\nu - method$ performs very well and converges to a solution in fewer iterations.

From this analysis we conclude that the $\nu - method$ shows the best combination of generalization performance and computational efficiency among the four regularization methods analyzed. We choose it as a representative for comparisons with other approaches. Table 6.3 compares the results obtained with the ν-method, with an SVM with RBF kernel, and also, for each dataset, with the classifier performing best among the 7 methods considered on the benchmark page (including RBF networks, Adaboost and Regularized AdaBoost, Kernel Fisher Discriminant, and SVMs with RBF kernels). The results obtained with the $\nu - method$ compare favorably with the ones achieved by the other methods.

6.1.2 Experiments on face detection

This section reports the analysis we carried out on the problem of face detection, to the purpose of evaluating the effectiveness of the ν-method in comparison to SVMs. The structure of the experiments, including model selection and error estimation, follows the one reported above. The data we consider are image patches, we represent them in the simplest way unfolding the patch matrix in a one-dimensional vector of integer values – the gray levels. All

Table 6.2: Comparison of the 5 methods we discuss. The average and standard deviation of the generalization error on the 13 datasets (numbered as in the Table 1) is reported on top and the value of the regularization parameter and the gaussian width - (t/σ) - on the bottom of each row. The best result for each dataset is in bold face.

	LANDWEBER	ν-METH	RLS	TSVD	IT($\nu = 5$)
1	11.70 ± 0.68	$\mathbf{10.67 \pm 0.53}$	11.22 ± 0.61	11.74 ± 0.63	10.96 ± 0.56
	(116/1)	(70/1)	(350/1)	(301/1)	(141/1)
2	25.38 ± 4.21	25.35 ± 4.24	$\mathbf{25.12 \pm 4.32}$	26.81 ± 4.32	25.26 ± 4.14
	(5/2)	(5/2)	(41/2)	(120/2)	(4/2)
3	23.70 ± 1.80	$\mathbf{23.60 \pm 1.82}$	24.40 ± 1.79	24.29 ± 0.2	23.63 ± 1.88
	(18/2)	(11/2)	(400/2)	(300/2)	(10/2)
4	34.27 ± 1.57	$\mathbf{34.25 \pm 1.59}$	34.31 ± 1.607	32.43 ± 0.90	30.92 ± 10.47
	(25/1)	(8/1)	(51/1)	(140/1)	(6/1)
5	23.20 ± 2.28	$\mathbf{23.14 \pm 2.34}$	23.37 ± 2.11	24.67 ± 2.60	23.31 ± 2.24
	(119/3)	(16/3)	(600/3)	(1150/3)	(51/3)
6	15.94 ± 3.37	$\mathbf{15.48 \pm 3.25}$	15.71 ± 3.20	15.58 ± 3.41	15.60 ± 3.41
	(63/12)	(16/12)	(500/12)	(170/12)	(21/12)
7	6.42 ± 0.82	2.78 ± 0.56	$\mathbf{2.68 \pm 0.54}$	2.99 ± 0.48	2.72 ± 0.53
	(7109/1)	(447/2.6)	(179000/2.6)	(280000/2.6)	(20001/2.6)
8	9.09 ± 0.89	3.09 ± 0.42	4.68 ± 0.7	$\mathbf{2.85 \pm 0.33}$	3.83 ± 0.52
	(514/3)	(37/3)	(820/3)	(510/3)	(151/3)
9	14.71 ± 0.75	$\mathbf{10.79 \pm 0.67}$	11.43 ± 0.72	11.67 ± 0.68	10.92 ± 0.72
	(816/6)	(72/6)	(1250/6)	(1400/6)	(501/6)
10	4.53 ± 2.34	4.55 ± 2.35	$\mathbf{4.48 \pm 2.33}$	4.49 ± 2.21	4.59 ± 2.34
	(65/1)	(28/1)	(100/1)	(200/1)	(21/1)
11	23.53 ± 1.82	22.96 ± 1.21	22.82 ± 1.81	$\mathbf{21.28 \pm 0.67}$	20.20 ± 7.17
	(5/1)	(1/1)	(1.19/1)	(12/1)	(1/1)
12	2.39 ± 0.13	$\mathbf{2.36 \pm 0.13}$	2.42 ± 0.14	2.39 ± 0.13	2.56 ± 0.30
	(20/3)	(7/3)	(100/3)	(61/3)	(1/3)
13	9.53 ± 0.45	9.63 ± 0.49	9.53 ± 0.44	9.77 ± 0.35	$\mathbf{9.52 \pm 0.44}$
	(8/3.1)	(12/3.1)	(150/3.1)	(171/3.1)	(21/3.1)

Table 6.3: Comparison of the ν-method (right column) against the best of the 7 methods taken from the benchmark webpage (see text) on the 13 benchmark datasets. The middle column shows the results for SVM from the same webpage.

	Best of 7	SVM	ν-meth.
Banana	LP_Reg-Ada		
	10.73 ± 0.43	11.53 ± 0.66	10.67 ± 0.53
B.Canc.	KFD		
	24.77 ± 4.63	26.04 ± 4.74	25.35 ± 4.24
Diabet.	KFD		
	23.21 ± 1.63	23.53 ± 1.73	23.60 ± 1.82
F.Solar	SVM-RBF		
	32.43 ± 1.82	32.43 ± 1.82	34.25 ± 1.59
German	SVM-RBF		
	23.61 ± 2.07	23.61 ± 2.07	23.14 ± 2.08
Heart	SVM-RBF		
	15.95 ± 3.26	15.95 ± 3.26	15.48 ± 3.25
Image	ADA_Reg		
	2.67 ± 0.61	2.96 ± 0.6	2.78 ± 0.56
Ringn.	ADA_Reg		
	1.58 ± 0.12	1.66 ± 0.2	3.09 ± 0.42
Splice	ADA_Reg		
	9.50 ± 0.65	10.88 ± 0.66	10.79 ± 0.67
Thyroid	KFD		
	4.20 ± 2.07	4.80 ± 2.19	4.55 ± 2.35
Titanic	SVM-RBF		
	22.42 ± 1.02	22.42 ± 1.02	22.96 ± 1.21
Twon.	KFD		
	2.61 ± 0.15	2.96 ± 0.23	2.36 ± 0.13
Wavef.	KFD		
	9.86 ± 0.44	9.88 ± 0.44	9.63 ± 0.49

#TRAIN + #TEST CLASSIFIER	600+1400	700+1300	800+1200
RBF-SVM	2.41 ± 1.39 $\sigma = 800$ $C = 1$	1.99 ± 0.82 $\sigma = 1000$ $C = 0.8$	1.60 ± 0.71 $\sigma = 1000$ $C = 0.8$
ν-method	$\mathbf{1.63 \pm 0.32}$ $\sigma = 341$ $t = 85$	$\mathbf{1.53 \pm 0.33}$ $\sigma = 341$ $t = 89$	$\mathbf{1.48 \pm 0.34}$ $\sigma = 300$ $t = 59$

Table 6.4: The data are the CBCL-MIT benchmark dataset of frontal faces (see text).

the images of the two datasets are 19×19, thus the size of our data is 361.

The first dataset we use for training and testing is the well known CBCL dataset for frontal faces[2] composed of thousands of small images of positive and negative examples of size. The face images obtained from this benchmark are clean and nicely registered.

The second dataset we consider is made of low quality images acquired by a monitoring system installed in our department[3]. The data are very different from the previous set since they have been obtained from video frames (therefore they are more noisy and often blurred by motion), faces have not been registered, gray values have not been normalized. The RBF kernel may take into account slight data misalignment due to the intraclass variability, but in this case model selection is more crucial and the choice of an appropriate parameter for the kernel is advisable.

The experiments performed on these two sets follow the structure discussed in the previous section. Starting from the original set of data, in both cases we randomly extract 2000 data that we use for most of our experiments: for a fixed training set size we generate 50 resamplings of training and test data. Then we vary the training set size from 600 (300+300) to 800 (400+400) training examples. The results obtained are reported in Table 6.4 and Table 6.5. The tables show a comparison between the ν-method and SVM as the size of the training set grows. The results obtained are slightly different: while on the CBCL dataset the ν-method performance is clearly above the SVM classifier, in the second set of data the performance of the ν-method increases as the training set size grows.

At the end of this evaluation process we retrained the ν-method on the whole set of 2000 data and again tuned the parameters with KCV obtaining $\sigma = 200$ and $t = 58$. Then we used this classifier to test a batch of newly acquired data (the size of this new test set is of 6000 images) obtaining a classification error of 3.67%. These results confirm the generalization ability of the algorithm. For completeness we report that the SVM classifier trained and tuned on the whole dataset of above — $\sigma = 600$ and $C = 1$ — lead to an error rate of 3.92%.

[2]Available for download at `http://cbcl.mit.edu/software-datasets/FaceData2.html`.
[3]The dataset is available upon request.

#TRAIN + #TEST CLASSIFIER	600+1400	700+1300	800+1200
RBF-SVM	**3.99 ± 1.21** $\sigma = 570$ $C = 2$	**3.90 ± 0.92** $\sigma = 550$ $C = 1$	3.8 ± 0.58 $\sigma = 550$ $C = 1$
ν-method	4.36 ± 0.53 $\sigma = 250$ $t = 67$	4.19 ± 0.50 $\sigma = 180$ $t = 39$	**3.69 ± 0.54** $\sigma = 200$ $t = 57$

Table 6.5: Average and standard deviation of the classification error of SVM and ν-method trained on training sets of increasing size. The data are a have been acquired by a monitoring system developed in our laboratory (see text).

6.2 Experimental results on annotation and retrieval

In this section we present a number of experiments conducted using the popular COREL30K database, which consists of 31.695 images from Corel Stock Photo CDs. The images are divided into 320 folders each representing a broad category ranging, for example, from *American National Parks*, to *Italian Cities*, from *Autumn*, to *Winter*, or *Mediterranean Cuisine* to *Dogs* or *Polar Bears*. Furthermore, each image has been labeled with up to 5 tags, from a set of 1036 keywords, most of which are just synonymous or specializations of the same general idea. We split the dataset in two parts; 28.000 images were used for learning and validation and the rest for independent tests. In order to make a thorough analysis of both positive and negative aspects of the modular algorithmic architecture described in the previous section, we conducted separated experiments focussing on all the specific modules.

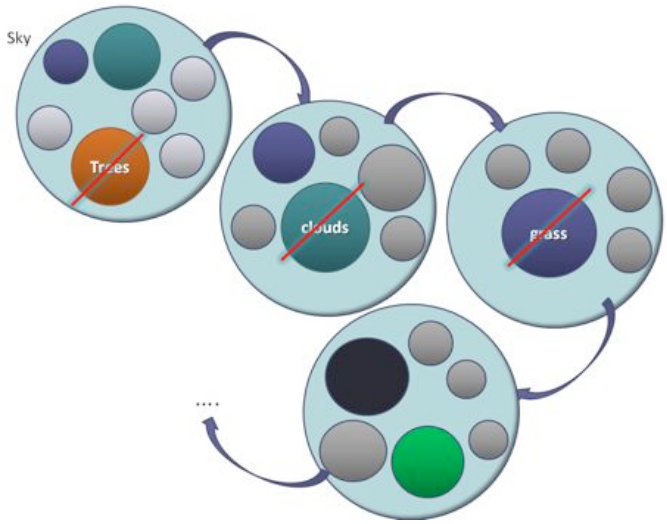
6.2.1 Identification of semantic classes of blobs

We start by describing the results of the algorithmic procedure discussed in section 4.4 to extract semantically relevant classes of blobs. The most difficult point is to group together image parts that are both similar from a visual viewpoint and likely to belong to the same semantic category. The critical issue is, of course, the fact that the words "semantic category" are extremely vague and it is difficult to translate such notion into an actual algorithm.
Our proposal is to separate the *categorization* (or clustering) of all the blobs in the dataset from the *identification* and selection of those specific clusters that are linked to a general category. In chapter 4 we discussed the actual clustering algorithm we designed to separate the blobs and outlined a method that uses the histograms of labels associated by human users to the whole images in order to transfer this prior information to the blobs. The main idea behind the method is to select all the clusters with a significantly different distribution of image tags. Actually, a simple direct implementation of this idea is able to identify only very general and broad categories, and some modifications are needed to make the procedure robust with respect to the errors and noise present in the clusters.
After some preliminary tests we specialized the idea and introduced an iterative procedure

that find out the general concepts from the entire dataset first. Therefore, all the blobs belonging to an identified concept are removed from the dataset and new, more specialized, concepts are retrieved. More specifically, the steps of the procedure are:

1. Find the class of blobs with the most different histogram from the one before the clustering.
2. Assign automatically a tentative tag to the semantic concept described by the tag associated to the bin of the histogram with the most significant difference.
3. Remove all the blobs contained in the above class.
4. Apply the clustering algorithm to the remaining blobs.
5. If there is at least one class with the same properties of that in the step 1, then repeat the steps 2 to 4.

The following figure, shows schematically the outlined procedure.



6.2.2 Examples of "visual properties" of blobs relative to the same specific concept

Figure 6.1 shows a number of randomly picked examples of blobs belonging to the same cluster. We have drawn the contour of the subregion in the original image from which the blob was extracted. Recall that the use three different cues to represent the information associated to the blob: texture, color and position. Since we balance the contribution of these three cues in the definition of similarity between blobs, at least in principle it is not possible to foresee

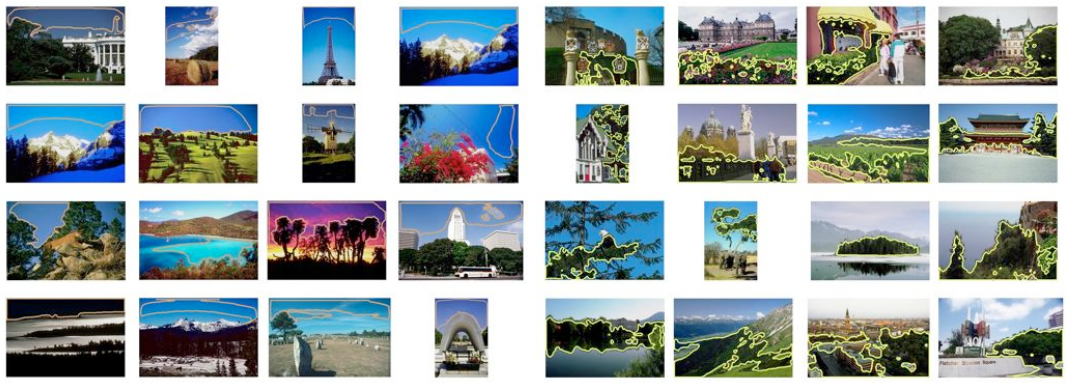
beforehand which cue is going to be dominant to discriminate one cluster form the others. In the examples below, we collected evidence that different situations are possible, depending on the distribution of the feature vectors in the database. For instance the second cluster – which a-posteriori can be associated to the concept *sky* – is certainly dominated by a strong color coherence, but also the position seems to play an important role. This is definitely true if one looks at the image at the position $(3, 3)$ in the grid of Figure 6.1(a) in which the blob of the sky has a strong red component, or the image at $(4, 1)$ in which the sky, correctly recognized, is black and only the position is a cue for determining its category. Examples of the cluster associated to the concept *trees* are displayed in Figure 6.1(b). In this case, the local texture of the blobs plays the most important role along with color, while the position is less characterizing. Again, color and texture are dominant cues also for the concept *grass* (see Figure 6.1(c)); however, here the scale of the texture is different from that of the trees. Due to such slight difference there may be errors and contamination between the two clusters such as those in Figures $(1, 2)$ and $(4, 1)$. In Figure 6.1(d), as it might be expectable, the concept snow is almost solely related to the presence of white regions, which can be easily confused with cloudy skies.

All the previous concepts have been selected during the first run of the iterative procedure outlined in the previous sub-section. Figures 6.1(f) and 6.1(e) show some images associated to concepts identified during subsequent runs. It is worth noting that either the kind of concepts becomes less general as the number of iterations increases or the visual properties of the blobs are less characterized.

In Figure 6.2.2 we show an example of a cluster whose associated histogram does not change significantly after the clustering process and that cannot be associated to a semantic concept in our schema. The local visual properties of the blobs are homogeneous but the blobs clearly belongs to semantically different concepts. Such cases can handled only by using to some extent the context of the whole image from which the blobs are extracted.

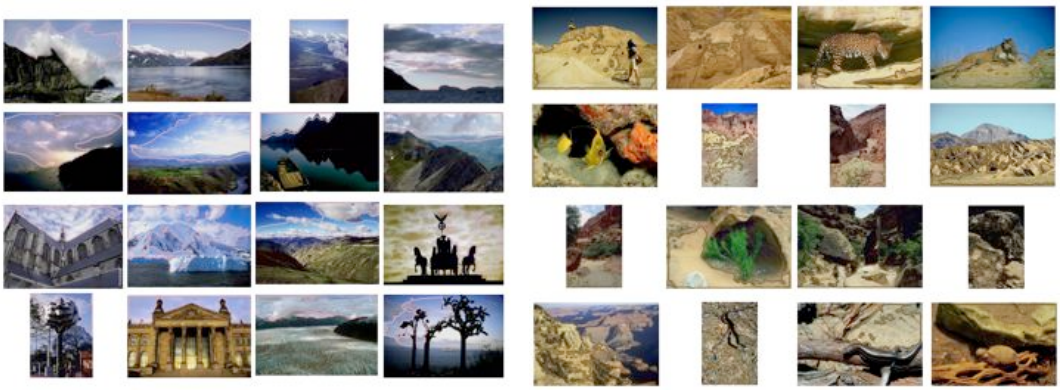
6.2.3 Performance evaluation of the concept-based classifiers trained using the ν-method.

In order to test the results obtained by the classifiers associated to specific semantic concepts, we adopted the standard approach of evaluating the *true* (TP) and *false* (FP) positive rates for different threshold values of the regression function learnt using the ν-method. Quantitatively, the measure of the effectiveness of a classifier can be evaluated through the ROC curve and the corresponding *area under the curve* (AUC).
We adopted the standard k-fold cross validation (KCV) learning strategy with $k = 5$ to prevent overfitting. Therefore, for each concept, we built 5 ROC curve and computed the AUC for the 5 training trials and them computed the mean and standard deviation across the trials. The generalization ability has been tested using a set of independent images that have not been used during the training stage. As positive training examples we used all the



(a) Sky (b) Trees

(c) Grass (d) Snow



(e) Clouds (f) Rocks

Figure 6.1: Examples of blobs associated to some specific semantic concept defined through the unsupervised procedure described in Section 4.1.

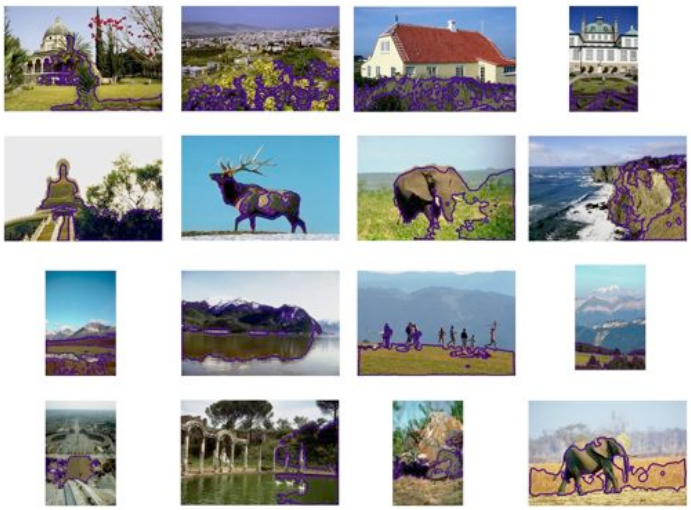


Figure 6.2: The above blobs belong to a cluster that can not be associated to a semantic class

blobs in the cluster associated with a concept, while for the negative examples we sampled randomly – with uniform probability – a set of blobs from images in all the other clusters. For positive and negative testing examples we annotated manually the blobs extracted from images in the test set discussed above. The corresponding AUCs are reported in the images below.
The performances obtained by the classifiers on different training splits are extremely high. As it was largely expected, the performances are less impressive when measured on previously unseen blobs. However, the AUCs remain always well above the 0.8 level which is generally considered the cutoff score to define effective classifiers.

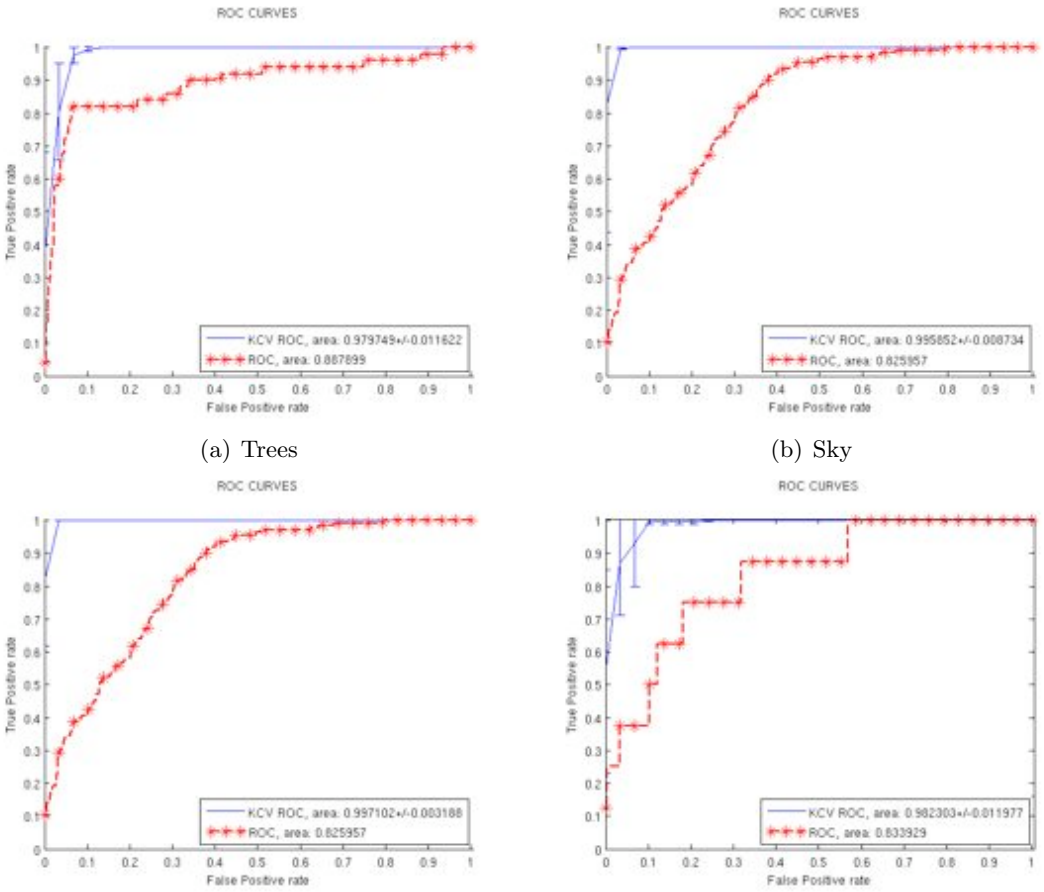
	SKY	TREES	GRASS	SNOW
n° IMAGES	225+225	340+340	613+613	272+272
KCV ERROR	0.017	0.010	0.043	0.05
PARAM.	$\lambda = 10$	$\lambda = 30$	$\lambda = 37$	$\lambda = 37$

Table 6.6: Performances of the concept-based classifiers for the largest classes that have been associated to semantic concept by our system. For each classifier, we reported both the cross validation error and the corresponding optimal regularization parameter.

The variability of the performances is mainly due to two different reasons. On the one hand, since currently we do not perform any post-processing to the clusters of blobs associated to a concept, it happens frequently that the noise level of the training labels is very high. In Figure 6.4 we show a few blobs which have been incorrectly labeled as positive trees examples by our unsupervised clustering procedure and used during the KCV training/validation procedure.

Our firm *trust* in the clustering process leads to an obvious discrepancy in the computation of TP and FP rates for train/validation and test stages. Therefore we believe the current comparison of the ROC curves is not completely fair. This point will be discussed more thoroughly below in this section.

On the other hand, the presence of performance differences between more specific concept (i.e. trees with higher generalization levels) and less specific concepts (i.e. sky) can be explained in terms of the possible presence of multiple *sub-concepts* such as cloudy vs. sunny skies, whose representative blobs are likely to be clustered in separate groups. Currently we do not make a specific check for this circumstance and do not combine such subgroups. This is a clear difference of conditions between training and test that somehow contributes to worsen the results of our system.



(a) Trees (b) Sky

(c) Grass (d) Snow

Figure 6.3: ROC curves obtained with the ν-method for the semantic concepts shown in Figure 6.1. The error bars refers to the k-fold cross validation procedure to avoid overfitting. The ROC curves are the average curves across the k-fold independent parameter learning.

A final important issue is the following: if we exploit the test set of the Corel dataset and we use the provided annotation, we add a further error to the noise that naturally appears in each "peak" cluster used for training the associated classifier. In Figure 6.5 we can observe the absence of the the tag trees for images including blobs.

(a) (b) (c) (d)

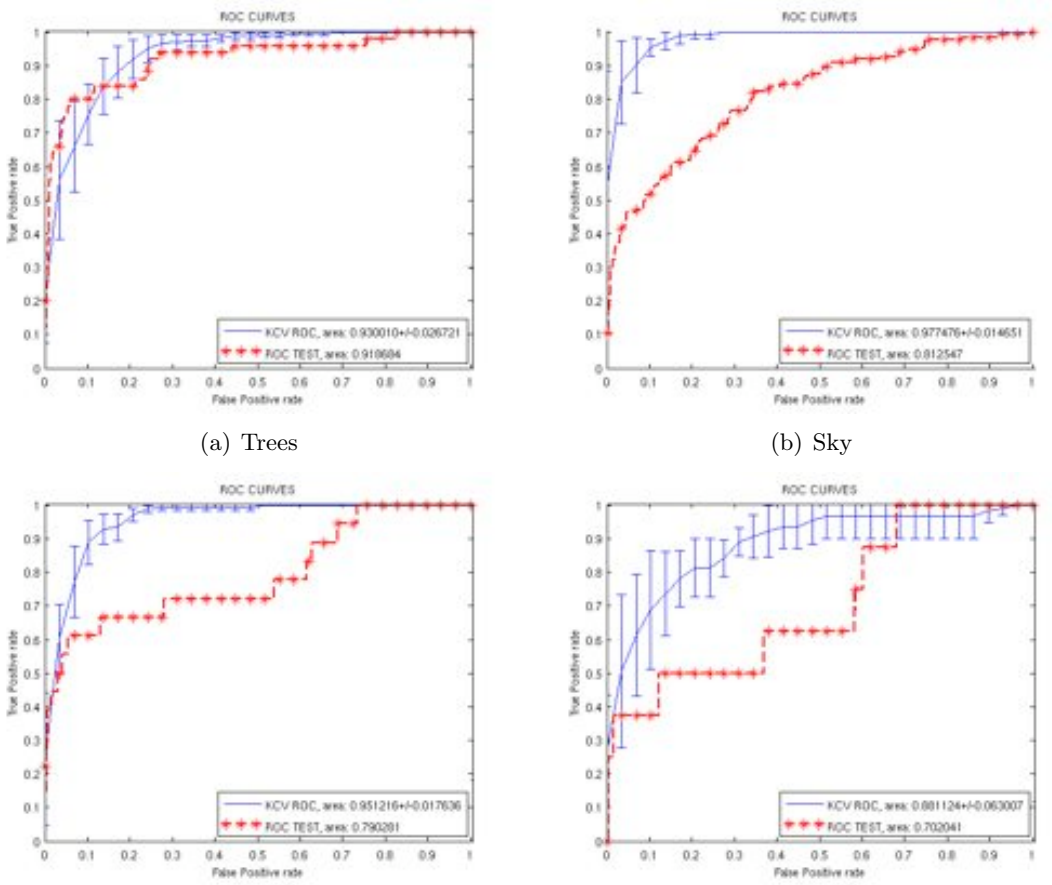
Figure 6.4: Examples of blobs incorrectly labeled as positive trees examples by our unsupervised clustering procedure to create the training set.

(a) (b) (c) (d)

Figure 6.5: Examples of images containing trees (positive examples in our training set) labeled in Corel in the following way: 6.5(a) mount, mountains, sky, vineyards 6.5(b) buildings, sky, village, water 6.5(c) elephants, sky, trunks, water 6.5(d) mountains, river, sky, water

Before concluding this part of the experiments, we addressed the issue of automatic labeling versus the manual labeling of training blobs. This provided us with a more principled way to evaluate the similarity between training and validation curves in the ROC figure. Indeed, we decided to select a selected number of concepts and annotated manually all the blobs before the training of the supervised classifier, without using the strong prior given by the results of the clustering.

Figures 6.6 show that the performances of the classifier both during training and validation are more comparable. Actually, the performances in the validation stage remained the almost same as in the previous case. However, the results obtained during the training stage are now more realistic given the large amount of noise present the images of Corel dataset.



(a) Trees (b) Sky

(c) Grass (d) Snow

Figure 6.6: ROC curves obtained with the ν-method with a training set annotated manually.

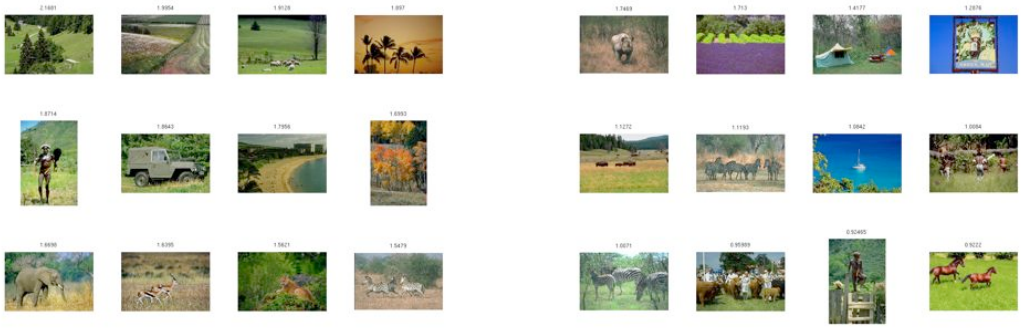
(a) nebulous (b) foggy (c) cloudy (d) clear

Figure 6.7: A fundamental issue that has not been addressed in our system is the presence of different subtypes of visual appearance within the same semantic concept. For instance, there are several examples of *sky* that are easily classified as the same by human user but can hardly considered as *similar* by an automatic algorithm that do not use too much prior and contextual information.

6.2.4 Results obtained by the retrieval system

In this section we present a number of examples of images retrieved by our system in correspondence to the following list of queries. Figures below show the ranking performance of our system by plotting the top 12 ranked images for different queries. Query modality that our system support is the "query by keyword" [DJL+08], i.e. we search a set of images containing a simply word corresponding to a concept learned in the training phase. In order to retrieve images we weight blobs by means of regression results of the concept classifier. More specifically, the ranking was performed using the sum of the regression labels of each blob for each image.

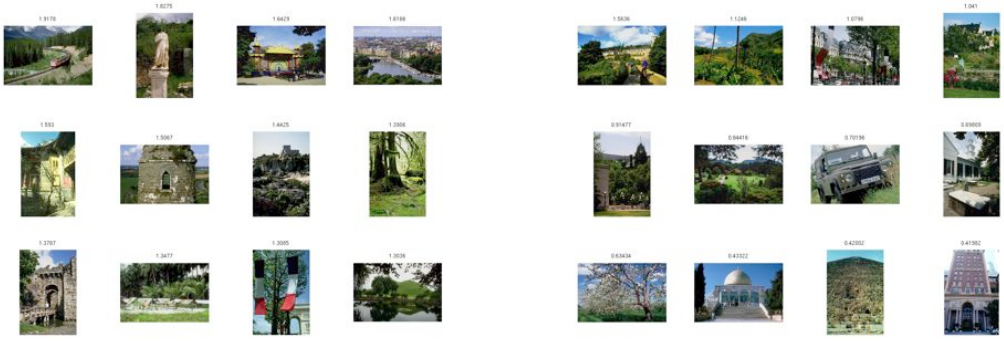
- *Query*: Top 12-ranked **grass**-images on two sub-samples of Corel.



- *Query*: Top 12-ranked **sky**-images on two sub-samples of Corel.

- *Query*: Top 12-ranked **snow**-images on two sub-samples of Corel.

- *Query*: Top 12-ranked **trees**-images on two sub-samples of Corel.



6.3 Distributed computing for feature extraction

Although the image segmentation of COREL30K can be computed just once, we may want to employ the overall system on a new dataset, or, more simply, to find concepts by means of a new set of features.

Computing the feature vectors for all dataset is computational expensive; the algorithm to retrieve features is made by four stages:

- transformation of color coordinates and application of the filter bank,
- application of kmeans algorithm to the color-texture measurements,

- computation of connected regions from clusters,
- information storage (segmentations and features).

The time used in the above stages for each image is about 3 to 6 seconds. Because of the sequential approach, the time for all the dataset is about 2 to 3 days. This module of the overall architecture is well-suited for a *distributed computing*, since each image processing is independent.

In distributed computing a program is split up into parts that run simultaneously on multiple computers communicating over a network. Distributed computing is a form of parallel computing, but parallel computing is most commonly used to describe program parts running simultaneously on multiple processors in the same computer. Both types of processing require dividing a program into parts that can run simultaneously, but distributed programs often must deal with heterogeneous environments, network links of varying latencies, and unpredictable failures in the network or the computers.

It can be costly and difficult to write programs so that they address concurrency issues. Actually, this algorithm can be adequately parallelized; a *grid* infrastructure can allow conventional, standalone programs to run on multiple machines. This eliminates computational costs or complications due to the concurrency: multiple instances of the same program running in the same shared memory and storage space at the same time. Such arrangements permit handling of data that would otherwise require the power of expensive supercomputers or would have been impossible to analyze.

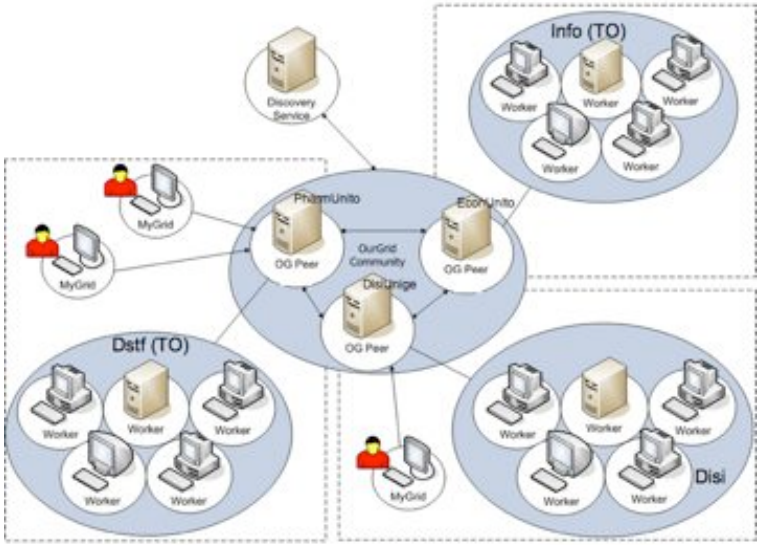


Figure 6.8: Grid architecture employed for extracting features from blobs.

In our analysis we use *OurGrid*[4] (see figure 6.8): a free-to-join peer-to-peer grid that has been in production since December 2004. Anyone can freely and easily join it to gain access to large amount of computational power and run parallel applications. This computational power is provided by the idle resources of all participants, and is shared in a way that give resources to those who contribute. Currently, the platform can be used to run any application whose *tasks* (i.e., parts that run on a single machine) do not communicate among themselves during execution, like most simulations, data mining and searching. All you need to connect to the grid is a linux machine.

We split COREL30K dataset into groups and we send to OurGrid the data, the OpenCv[5] libraries and the program that computes the four stages mentioned above (this ingredients identify a task). In the perspective of extending our architecture to new concepts and data benefits given by a parallel computation are fundamental: for COREL30K dataset the computational costs are reduced to few hours (2 or 3).

[4]http://www.ourgrid.org/

[5]OpenCV is a computer vision library originally developed by Intel (see http://www710.univ-lyon1.fr/ bouakaz/OpenCV-0.9.5/docs/).

Chapter 7

Conclusions

In this thesis we presented our work on automatic image annotation based on learning visual cues. In order to cope with the difficulties concerning the development of an annotation and retrieval system, we tried to design a modular architecture. Specifically, we focused on the learning aspects because they represent an essential part of the original contribution of the work described in the thesis.

This work involves two different learning approaches, applied respectively in two different modules of the algorithmic pipeline. Therefore, the contributions of this thesis can be summarized as follows:

- the development of an *unsupervised learning* strategy to organize data in homogeneous clusters and to label automatically regions belonging to the most representative clusters,
- the development of *supervised learning* algorithms able to generalize in a simple and efficient way semantic concepts obtained at the previous step.

With regard to the development of an unsupervised strategy (see Chapter 4), we tried to overcome the weaknesses of several current CBIR systems attempting to exploit the annotation given by users relative to *entire images* and the *perceptual similarity* between blobs. The set of tags were not fixed a priori from a dictionary of popular keywords, but rather they were extracted automatically from an initial collection of images by means of unsupervised learning techniques which allow for an univocal and algorithmic definition of *semantic visual concepts*. This makes the system incrementally extendable to new semantic concepts.
Another advantage of our approach is that our system needs only *weak prior knowledge* from human users. For example, in the experiments described in Chapter 6, users were asked to associate one or more textual tag to the images of the database. In order to appreciate the advantages of the proposed idea, let's recall that – since our approach is region-based – a fully supervised approach would require complete annotation of all the segments and sub-regions

of the images in the database. As we argued in Chapter 1, this is an important point because the constraint of fully annotation is one of the trickiest issues for a CBIR system designer, and often it is the actual system bottleneck leading to a lack of generalization and to the impossibility of an efficient extension to new concepts (i.e. those who are not modeled at design time).

With regard to supervised learning (see Chapter 5) we developed algorithms representing a suitable alternative, in terms of generalization performances and computational efficiency, to state-of-the-art algorithms for classification, like SVM and adaboost. The performance of these spectral algorithms against state-of-the-art techniques, such as SVMs, has been assessed on various datasets (see Chapter 6). One of the main advantages of the methods we proposed is their simplicity: each spectral algorithm is an easy-to-use linear method whose implementation is straightforward. Indeed our experience suggests that this helps dealing with overfitting in a transparent way and makes the model selection step easier. In particular, the search for the best choice of the regularization parameter in iterative schemes is naturally embedded in the iteration procedure.

By taking advantage on ν-method – the algorithm with better performances and computational properties – our system for automatic image annotation and retrieval has been used for solving two subproblems within the context of CBIR: the automatic annotation of blobs belonging to the training sets and the retrieval task.

The experimental validation discussed in Chapter 6 confirms the potential of the proposed approach but it deserves more investigation and improvements.
Future developments will include:

- the generalization of the *conceptualization* stage using a multi-class learning approach instead of a single class,
- the integration within the system of algorithms for the automatic selection of the visual features and
- the introduction of context information.

Bridging the cognitive gap in image retrieval is still an active research. We think that trying to solve the annotation problem with a data-driven procedure is the right way, but it implies obstacles. In fact, in our approach, some concepts are better "represented" in terms of tags by the employed annotated dataset. On the other hand, some classes are less explicit or do not represent semantic human concepts. Our procedure makes difficult to find more than few concepts or, more specifically, is able to find a satisfactory number of sub-concepts. Currently, this is a drawback of our architecture and we hope that, with the proposed future developments, it will be overcome.

Bibliography

[A.10] Haar A. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69:331–371, 1910.

[Add02] Paul S. Addison. *The Illustrated Wavelet Transform Handbook*. Institute of Physics, 2002. ISBN 0-7503-0692-0.

[AIS93] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *In Proc. of ACM SIGMOD International Conference on Management of Data*, 1993.

[Aro50] N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68:337–404, 1950.

[BB98] M. Bertero and P. Boccacci. *Introduction to Inverse Problems in Imaging*. IOP Publishing, Bristol, 1998.

[BBP00] Stefano Berretti, Alberto Del Bimbo, and Pietro Pala. Retrieval by shape similarity with perceptual distance and effective indexing. *IEEE Transactions on Multimedia*, 2:225–239, 2000.

[BCP05] Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. Warp: Accurate retrieval of shapes using phase of fourier descriptors and time warping distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):142–147, 2005.

[BE02] O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

[BF77] C.R. Brice and C.L. Fennema. Scene analysis using regions. In *CMetImAly77*, pages 79–100, 1977.

[BG05] I. Borg and P. J. F. Groenen. *Modern Multidimensional Scaling: Theory and Applications (Springer Series in Statistics)*. Springer, Berlin, 2nd ed. edition, September 2005.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[BJM06] Peter L. Bartlett, Michael I. Jordan, and Jon D. McAuliffe. Convexity, classification, and risk bounds. *J. Amer. Statist. Assoc.*, 101(473):138–156, 2006.

[BL76] Ruzena Bajcsy and Lawrence Lieberman. Texture gradient as a depth cue. 5(1):52–67, March 1976. BAJESY76a.

[BLL89] R. Bajcsy, S. W. Lee, and A. Leonardis. Image segmentation with detection of highlights and inter-reflections using color. Technical report, June 1989.

[BMP02] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[BN02] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, volume 14, pages 585–591, 2002.

[Bou02] Olivier Bousquet. *Concentration Inequalities and Empirical Processes Theory Applied to the Analysis of Learning Algorithms*. PhD thesis, Ecole Polytechnique, 2002.

[BPR06] F. Bauer, S. Pereverzev, , and L. Rosasco. On regularization algorithms in learning theory. *Journal of Complexity,*, 2006. In Press, doi: 10.1016/j.jco.2006.07.001 , Online 19 October 2006,.

[Bur98] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[BY02] P. Bühlmann and B. Yu. Boosting with the l_2-loss: Regression and classification. *Journal of American Statistical Association*, 98:324–340, 2002.

[Cap06] A. Caponnetto. Optimal rates for regularization operators in learning theory. Technical report, CBCL Paper #264/ CSAIL-TR #2006-062, M.I.T, 2006. available at http://cbcl.mit.edu/projects/cbcl/publications/ps/MIT-CSAIL-TR-2006-062.pdf.

[CCMV07] G. Carneiro, A.B. Chan, P.J. Moreno, and N. Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(3):394–410, 2007.

[CD05] David A. Clausi and Huang Deng. Design-based texture feature fusion using gabor filters and co-occurrence probabilities. *IEEE Transactions on Image Processing*, 14(7):925–936, 2005.

[CDV06] A. Caponnetto and E. De Vito. Optimal rates for regularized least-squares algorithm. *Found. Comput. Math.*, 2006. In Press, DOI 10.1007/s10208-006-0196-8, Online August 2006.

[Chu97] O. Chung. Spectral graph theory (reprinted with corrections). In *CBMS: Conference Board of the Mathematical Sciences, Regional Conference Series*, 1997.

[CL06] Ronald R. Coifman and Stephane Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, July 2006.

[Com94] Pierre Comon. Independent component analysis, a new concept? *Signal Process.*, 36(3):287–314, 1994.

[CPZ97] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 426–435, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[CST00] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge Univ. Press, 2000.

[CWS03] O. Chapelle, J. Weston, and B. Scholkopf. Cluster kernels for semi-supervised learning. In *Neural Information Processing Systems 15*, pages 585–592, 2003.

[Dau88] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 41(7):909–996, 1988.

[Dau92] I. Daubechies. *Ten Lectures on Wavelets.* Society for Industrial and Applied Mathematics, 1992. ISBN 0-89871-274-2.

[DCA79] Larry S. Davis, M. Clearman, and J. K. Aggarwal. A comparative texture classification study based on generalized cooccurrence matrices. In *in IEEE Conf. on Decision and Control*, pages 12–14. Springer-Verlag, 1979.

[DH95] D. Dunn and W.E. Higgins. Optimal gabor filters for texture segmentation. *Image Processing, IEEE Transactions on*, 4(7):947–964, Jul 1995.

[DJL+08] Ritendra Datta, Dhiraj Joshi, Jia Li, James, and Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2), 2008.

[DV02] Minh N. Do and Martin Vetterli. Wavelet-based texture retrieval using generalized gaussian density and kullback-leibler distance. *IEEE Trans. Image Processing*, 11:146–158, 2002.

[DVCR05] E. De Vito, A. Caponnetto, and L. Rosasco. Model selection for regularized least-squares algorithm in learning theory. *Found. Comput. Math.*, 5(1):59–85, 2005.

[DVRC+05] E. De Vito, L. Rosasco, A. Caponnetto, U. De Giovannini, and F. Odone. Learning from examples as an inverse problem. *Journal of Machine Learning Research*, 6:883–904, May 2005.

[DVRC06] Ernesto De Vito, Lorenzo Rosasco, and Andrea Caponnetto. Discretization error analysis for Tikhonov regularization. *Anal. Appl.*, 4(1):81–99, 2006.

[DVRV05] E. De Vito, L. Rosasco, and A. Verri. Spectral methods for regularization in learning theory. Technical Report Technical Rerport, DISI, Universitá degli Studi di Genova, Italy, 2005.

[Efr87] Bradley Efron. *The Jackknife, the Bootstrap, and Other Resampling Plans (CBMS-NSF Regional Conference Series in Applied Mathematics)*. Society for Industrial & Applied Mathematics, January 1987.

[EHN96] H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of inverse problems*, volume 375. Kluwer Academic Publishers Group, Dordrecht, 1996.

[ELL01] Brian S. Everitt, Sabine Landau, and Morven Leese. *Cluster Analysis*. Arnold Publishers, May 2001.

[EPP00a] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

[EPP00b] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Adv. Comp. Math.*, 13:1–50, 2000.

[FSN+95] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yanker. Query by image and video content: The qbic system. *Computer*, 28(9):23–32, 1995.

[FT74] J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *Computers, IEEE Transactions on*, C-23(9):881–890, 1974.

[Gab46] D. Gabor. Theory of communication. *Proc. of IEEE*, 93:429–457, 1946.

[GD03] Guodong Guo and C.R. Dyer. Simultaneous feature selection and classifier training via linear programming: a case study for face expression recognition. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 1:I–346–I–352, 18-20 June 2003.

[GJP95] Federico Girosi, Michael Jones, and Tomaso Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

[GKRR01] Roman Goldenberg, Ron Kimmel, Ehud Rivlin, and Michael Rudzsky. Fast geodesic active contours, 2001.

[GvdBSG01] J.M. Geusebroek, R. van den Boomgaard, A.W.M. Smeulders, and H. Geerts. Color invariance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1338–1350, 2001.

[GVL96] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[GW08] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Pub (Sd), 3 edition, May 2008.

[Had02] J. Hadamard. Sur les problèmes aux dériveés partielles et leur signification physique. In *Princeton University Bulletin*, number 23, pages 49–52, 1902.

[Had23] J. Hadamard. *Lectures on the Cauchy Problem in Linear Partial Differential Equations*. Yale University Press, 1923.

[Has84] T.J. Hastie. Principal curves and surfaces. *Laboratory for Computational Statistics Technical Report 11, Stanford University, Dept. of Statistics*, 1984.

[HGN04] Efstathios Hadjidemetriou, Michael D. Grossberg, and Shree K. Nayar. Multiresolution histograms and their use for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7):831–847, 2004.

[HGS05] Minh A. Hoang, Jan-Mark Geusebroek, and Arnold W. M. Smeulders. Color texture measurement and segmentation. *Signal Process.*, 85(2):265–275, 2005.

[HKM+97] Jing Huang, S. Ravi Kumar, Mandar Mitra, Wei-Jing Zhu, and Ramin Zabih. Image indexing using color correlograms. In *CVPR 1997: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR 1997)*, page 762, Washington, DC, USA, 1997. IEEE Computer Society.

[HP76] S.L. Horowitz and T. Pavlidis. Picture segmentation by a tree traversal algorithm. *Journal of the ACM*, 23(2):368–388, April 1976.

[HSD73] Robert M. Haralick, K. Shanmugam, and Its'hak Dinstein. Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*, 3(6):610–621, 1973.

[HTF01] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, August 2001.

[HTZ04] S. Hastie, T. andRosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *JMLR*, 5:1391–1415, 2004.

[JD88] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[JF90] A. K. Jain and F. Farrokhnia. Unsupervised texture segmentation using gabor filters. In *Systems, Man and Cybernetics, 1990. Conference Proceedings., IEEE International Conference on*, pages 14–19, 1990.

[JF91] Anil K. Jain and Farshid Farrokhnia. Unsupervised texture segmentation using gabor filters. 24(12), 1991.

[JRL97] Anil K. Jain, Nalini K. Ratha, and Sridhar Lakshmanan. Object detection using gabor filters. *Pattern Recognition*, 30(2):295–309, February 1997.

[Jul81] Bela Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, March 1981.

[KBC04] Manesh Kokare, P. K. Biswas, and B. N. Chatterji. Rotated complex wavelet based texture features for content based image retrieval. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 1*, pages 652–655, Washington, DC, USA, 2004. IEEE Computer Society.

[Koh90] T. Kohonen. The self-organizing map. In *Proceedings of the IEEE*, volume 78, pages 1464–1480, 1990.

[Kol01] Erica Kolatch. Clustering algorithms for spatial databases: A survey, 2001.

[KSK90] Gudrun J. Klinker, Steven A. Shafer, and Takeo Kanade. A physical approach to color image understanding. *International Journal of Computer Vision*, 4:7–38, 1990.

[KWT88] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.

[LGRO+08] L. Lo Gerfo, L. Rosasco, F. Odone, E. De Vito, and A. Verri. Spectral algorithms for supervised learning. *Neural Comput.*, 20(7):1873–1897, 2008.

[LL00] Longin Jan Latecki and Rolf Lakämper. Shape similarity measure based on correspondence of visual parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1185–1190, 2000.

[LLL+02] Yi Lin, Yi Lin, Yoonkyung Lee, Yoonkyung Lee, Grace Wahba, and Grace Wahba. Support vector machines for classification in nonstandard situations. In *Machine Learning*, pages 191–202, 2002.

[LM01] L. Lucchese and S. K. Mitra. Color image segmentation: A state-of-the-art survey, 2001.

[Mac67] J. Macqueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1 of *Berkeley: University of California Press*, pages 281–297, 1967.

[Mah36] P. C. Mahalanobis. On the generalized distance in statistics. pages 49–55, 1936.

[Mal89] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(7):674–693, July 1989.

[Man96] B. S. Manjunath. Texture features for browsing and retrieval of image data. *IEEE Trans Pattern Anal Mach Intell*, 18(8):837–842, 1996.

[MC93] B. S. Manjunath and R. Chellappa. A unified approach to boundary perception: edges, textures and illusory contours. *IEEE Trans. Neural Networks*, 4(1):96–108, Jan 1993.

[MG95] Rajiv Mehrotra and James E. Gary. Similar-shape retrieval in shape data management. *Computer*, 28(9):57–62, 1995.

[MrOVY01] B. S. Manjunath, Jens rainer Ohm, Vinod V. Vasudevan, and Akio Yamada. Color and texture descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11:703–715, 2001.

[MXH06] C. A. Micchelli, Y. Xu, and Zhang H. Universal kernels. *JMLR*, 7:2651–2667, 2006.

[NBA93] W. Niblack, R. Barber, and Al. The qbic project: Queying images by content using color, texture and shape. In *International Symposium on Electronic Imaging: Science and Technology, in Storage and Retrieval for Image and Video Database*, volume 1908, February 1993.

[NJW02] A.Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS 14*, 2002.

[OBV05] F. Odone, A. Barla, and A. Verri. Building kernels from binary strings for image matching. *Image Processing, IEEE Transactions on*, 14:169–180, 2005.

[OS88] Stanley Osher and James A. Sethian. Fronts propagating with curvature dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[PBE+06] Jean Ponce, T. L. Berg, M. Everingham, D. Forsyth, M. Hebert, Svetlana Lazebnik, Marcin Marszałek, Cordelia Schmid, C. Russell, A. Torralba, C. Williams, Jianguo Zhang, and Andrew Zisserman. Dataset issues in object recognition. In *Towards Category-Level Object Recognition*, pages 29–48. Springer, 2006.

[PDM02] Euripides G.M. Petrakis, Aristeidis Diplaros, and Evangelos Milios. Matching and retrieval of distorted and occluded shapes using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11):1501–1516, 2002.

[Pea01] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.

[PG92] T. Poggio and F. Girosi. A theory of networks for approximation and learning. In C. Lau, editor, *Foundation of Neural Networks*, pages 91–106. IEEE Press, Piscataway, N.J., 1992.

[PRMN04] T. Poggio, R. Rifkin, S. Mukherjee, and P. Niyogi. General conditions for predictivity in le arning theory. *Nature*, 428:419–422, 2004.

[PRTB99] J. Puzicha, Y. Rubner, C. Tomasi, and J. Buhmann. Empirical evaluation of dissimilarity measures for color and texture. In *IEEE International Conference on Computer Vision*, pages 1165–1173, September 1999.

[PV98] M. Pontil and A. Verri. Properties of support vector machines. *Neural Computation*, 10:977–996, 1998.

[PZ96] G. Pass and R. Zabih. Histogram refinement for content-based image retrieval. In *WACV '96: Proceedings of the 3rd IEEE Workshop on Applications of Computer Vision (WACV '96)*, page 96, Washington, DC, USA, 1996. IEEE Computer Society.

[RH99] Trygve Randen and John Hakon Husøy. Filtering for texture classification: A comparative study. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(291–310):4, 1999.

[RHC99] Y. Rui, T. Huang, and S. Chang. Image retrieval: current techniques, promising directions and open issues, # apr 1999.

[RMP05] A. Rakhlin, S. Mukherjee, and T. Poggio. Stability results in learning theory. *Analysis and Applications*, 3:397–419, 2005.

[ROD01] David G. Stork Richard O. Duda, Peter E. Hart. *Pattern Recognition and Machine Learning*. Wiley, 2001, 2001.

[Ros06] Lorenzo Rosasco. *Regularization Approaches in Learning Theory.* PhD thesis, Dipartimento di Informatica e Scienza dell'Informazione – Universita degli Studi di Genova (DISI), 2006.

[RS00] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 2000.

[RTG98] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A metric for distributions with applications to image databases. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, Washington, DC, USA, 1998. IEEE Computer Society.

[RTG00] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth moverś distance as a metric for image retrieval. *International Journal of Computer Vision*, 40:2000, 2000.

[RV08] N. Rasiwasia and N. Vasconcelos. A study of query by semantic example. In *Semantic Learning Applications in Multimedia*, pages 1–8, 2008.

[SB91] Michael J. Swain and Dana H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, November 1991.

[SC96] John R. Smith and Shih F. Chang. Tools and techniques for color image retrieval. In *SPIE Storage and Retrieval for Image and Video Databases IV*, volume 2670. International Society for Optical Engineering, 1996.

[SD96] Markus Stricker and Er Dimai. Color indexing with weak spatial constraints, 1996.

[SDS95] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. Wavelets for computer graphics: A primer. *IEEE Computer Graphics and Applications*, 15(3):76–84, 1995.

[Ser82] J. Serra. *Image Analysis and Mathematical Morphology*, volume 1. Academic Press, London, England, 1982.

[SK94] Wladyslaw Skarbek and Andreas Koschan. Colour image segmentation — a survey. Technical report, Institute for Technical Informatics, Technical University of Berlin, October 1994.

[SK03] A. Smola and R. Kondor. Kernels and regularization on graphs. In *COLT*, 2003.

[SM00] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8), 2000.

[SS02] B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[STC04] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, June 2004.

[SWS+00] 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. Mach. Intell.*, 22(12):1349–1380, 2000.

[SZ04] Steve Smale and Ding-Xuan Zhou. Shannon sampling and function reconstruction from point values. *Bull. Amer. Math. Soc. (N.S.)*, 41(3):279–305 (electronic), 2004.

[SZ05a] S. Smale and D.X Zhou. Learning theory estimates via integral operators and their approximations. Technical report, Toyota Technological Institute at Chicago, USA, 2005. available at http://ttic.uchicago.edu/ smale/papers/sampIII5412.pdf.

[SZ05b] Steve Smale and Ding-Xuan Zhou. Shannon sampling. II. Connections to learning theory. *Appl. Comput. Harmon. Anal.*, 19(3):285–302, 2005.

[TA77] A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill Posed Problems*. W. H. Winston, Washington, D.C., 1977.

[TdSL00] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290, 2000.

[TJ90] Mihran Tuceryan and Anil K. Jain. Texture segmentation using voronoi polygons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:211–216, 1990.

[TJ98] Mihran Tuceryan and Anil K. Jain. *The Handbook of Pattern Recognition and Computer Vision*, chapter Texture Analysis, pages 207–248. World Scientific Publishing Co., 2 edition, 1998.

[TT90] Fumiaki Tomita and Saburo Tsuji. *Computer Analysis of Visual Textures*. Kluwer Academic Publishers, Norwell, MA, USA, 1990.

[Vap82] Vladimir Vapnik. *Estimation of dependences based on empirical data*. Springer Series in Statistics. Springer-Verlag, New York, 1982. Translated from the Russian by Samuel Kotz.

[Vap98] V. Vapnik. *Statistical learning theory*. John Wiley and sons, New York, 1998.

[VP87] H. Voorhees and T. Poggio. Detecting textons and texture boundaries in natural images. *ICCV*, 87:250–258, 1987.

[Wah90] G. Wahba. *Spline models for observational data*, volume 59. SIAM, Philadelphia, PA, 1990.

[WBB+06] James Z. Wang, Nozha Boujemaa, Alberto Del Bimbo, Donald Geman, Alexander G. Hauptmann, and Jelena Tesić. Diversity in multimedia information retrieval research. In *MIR '06: Proceedings of the 8th ACM international workshop on Multimedia information retrieval*, pages 5–12, New York, NY, USA, 2006. ACM.

[WDR76] J.S. Weszka, C.R. Dyer, and A. Rosenfeld. A comparative study of texture measures for terrain classification. *IEEE Trans SMC*, 6:269–285, 1976.

[WYZ06] Qiang Wu, Yiming Ying, and Ding-Xuan Zhou. Learning rates of least-square regularized regression. *Found. Comput. Math.*, 6(2):171–192, 2006.

[YRC07] Y. Yao, L. Rosasco, and A. Caponnetto. On early stopping in gradient descent learning,. *Constructive Approximation*, 2007. In Press, available at http://math.berkeley.edu/ yao/publications/earlystop.pdf.

[YWB74] Ian T. Young, Joseph E. Walker, and Jack E. Bowie. An analysis technique for biological shape. i. *Information and Control*, 25(4):357 – 370, 1974.

[ZA06] Tong Zhang and Rie Ando. Analysis of spectral kernel design based semi-supervised learning. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1601–1608. MIT Press, Cambridge, MA, 2006.

[ZKG+05] X. Zhu, J. Kandola, Z. Ghahramani, , and J. Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. In *Neural Information Processing Systems 17*, 2005.

[ZL04] D. Zhang and G. Lu. Review of shape representation and description techniques. *Pattern Recognition*, 37(1):1–19, 2004.

[ZWIL00] Dengsheng Zhang, Aylwin Wong, Maria Indrawan, and Guojun Lu. Content-based image retrieval using gabor texture features. In *IEEE Transactions PAMI*, pages 13–15, 2000.