Universitat
Autònoma
de Barcelona

Probabilistic Darwin Machines:

A new approach to develop Evolutionary Object Detection Systems

A dissertation submitted by **Xavier Baró i Solé** at Universitat Autònoma de Barcelona to fulfil the degree of **Doctor en Informàtica**.

Bellaterra, February 2009

Advisor: **Dr. Jordi Vitrià i Marca**
Dept. Matemàtica Aplicada i Anàlisi, Universitat de Barcelona.
Computer Vision Center, Universitat Autònoma de Barcelona.

Centre de Visió per Computador

This document was typeset by the author using LaTeX 2_ε.

The research described in this book was carried out at the Computer Vision Center, Universitat Autònoma de Barcelona.

ISBN 978–84–936529–2–0

Printed by Ediciones Gráficas Rey, S.L.

A na Júlia i en Sergi.

Acknowledgment

Primer de tot voldria agrair al meu director de tesi Jordi Vitrià l'oportunitat de poder fer una tesi, i l'ajuda rebuda durant la seva realització, tant per la seva capacitat per tenir sempre dues referències per a cada pregunta, com una alternativa per a cada obstacle. La realització de la tesi tampoc no hauria estat possible sense el suport de na Petia Radeva i en Sergio Escalera, part incansable d'aquest projecte des del seu inici.

També m'agradaria agrair a la gent de l'*Institut Cartogràfic de Catalunya* i molt especialment a na Maria Pla, per la seva col·laboració, tant a nivell tècnic com personal. El poder treballar en problemes i dades reals, junt amb els seus comentaris i suggeriments al llarg d'aquests anys han permès donar una visió més pràctica als mètodes proposats, així com detectar i corregir algunes limitacions en la seva aplicació real.

Igualment agrair a tota la família que, tot i no entendre moltes vegades de què anava tot el tema, ha donat el suport necessari, entenent que les visites cada cop es reduïssin més. També als amics de la universitat, la meva altra família: Mireia, Ricky, Hanna, Vicente, Guillem i tants altres, amb els quals després de compartir tota una carrera continuen aquí, creant un espai on desconnectar i sentir-se com a casa quan és necessari.

Després de tants anys de "barbacoes", karts, futbol, sopars, festes de tot tipus i, perquè no dir-ho, també hem assistit a algun que altre seminari. Per tot això i molt més, no podria deixar-me a tota la gent del CVC, tant els que hi són com els que hi han estat, els diferents grups que formen aquest ecosistema tan especial: Raquel, Àgata, Agnès, Carme, Enric, Jaume, Mireia, Raul, Ana Maria, ... Entre tots heu aconseguit fer del lloc de treball un espai agradable, que sí Jose, que el CVC té les seves coses bones, que fins i tot tenim el nostre propi cava, elaborat amb tota la il·lusió del món, i amb la guia del nostre sommelier i mestre de caves i beuratges Xavi Roca.

Finalment, i no per això menys important, donar les gràcies als integrants dels dinars dels dimarts, Sergio i Poal, per crear un espai de debat transcendental (o no tant), on tots els problemes semblen solucionar-se o com a mínim diluir-se durant un parell d'horetes. Tampoc no vull oblidar les esporàdiques intervencions, a vegades estel·lars de les "nenes": Mari, Aura, Alícia, Debora,... les quals vénen a posar els "serrells entre contenidors", enterrant tots els tòpics, haguts i per haver.

This work was supported in part by a research grant from projects TIN2006-15308-C02 and CONSOLIDER-INGENIO 2010 (CSD2007-00018).

Abstract

Ever since computers were invented, we have wondered whether they might perform some of the human quotidian tasks. One of the most studied and still nowadays less understood problem is the capacity to learn from our experiences and how we generalize the knowledge that we acquire.

One of that unaware tasks for the persons and that more interest is awakening in different scientific areas since the beginning, is the one that is known as pattern recognition. The creation of models that represent the world that surrounds us, help us for recognizing objects in our environment, to predict situations, to identify behaviors... All this information allows us to adapt ourselves and to interact with our environment. The capacity of adaptation of individuals to their environment has been related to the amount of patterns that are capable of identifying.

When we speak about pattern recognition in the field of Computer Vision, we refer to the ability to identify objects using the information contained in one or more images. Although the progress in the last years, and the fact that nowadays we are already able to obtain "useful" results in real environments, we are still very far from having a system with the same capacity of abstraction and robustness as the human visual system.

In this thesis, the face detector of Viola & Jones is studied as the paradigmatic and most extended approach to the object detection problem. Firstly, we analyze the way to describe the objects using comparisons of the illumination values in adjacent zones of the images, and how this information is organized later to create more complex structures. As a result of this study, two weak points are identified in this family of methods: The first makes reference to the description of the objects, and the second is a limitation of the learning algorithm, which hampers the utilization of best descriptors.

Describing objects using Haar-like features limits the extracted information to connected regions of the object. In the case we want to compare distant zones, large contiguous regions must be used, which provokes that the obtained values depend more on the average of lighting values of the object than in the regions we are wanted to compare. With the goal to be able to use this type of non local information, we introduce the Dissociated Dipoles into the outline of objects detection.

The problem using this type of descriptors is that the great cardinality of this feature set makes unfeasible the use of Adaboost as learning algorithm. The reason is that during the learning process, an exhaustive search is made over the space of hypotheses, and since it is enormous, the necessary time for learning becomes prohibitive. Although we studied this phenomenon on the Viola & Jones approach, it is a general problem for most of the approaches, where learning methods introduce a limitation on the descriptors that can be used, and therefore, on the quality of the object description. In order to remove this limitation, we introduce evolutionary methods into the Adaboost algorithm, studying the effects of this modification on the learning ability. Our experiments conclude that not only it continues being able to learn, but its convergence speed is not significantly altered.

This new Adaboost with evolutionary strategies opens the door to the use of feature sets with an arbitrary cardinality, which allows us to investigate new ways to describe our objects, such as the use of Dissociated Dipoles. We first compare the learning ability of this evolutionary Adaboost using Haar-like features and Dissociated Dipoles, and from the results of this comparison, we conclude that both types of descriptors have similar representation power, but depends on the problem they are applied, one adapts a little better than the other. With the aim of obtaining a descriptor capable of share the strong points from both Haar-like and Dissociated Dipoles, we propose a new type of feature, the Weighted Dissociated Dipoles, which combines the robustness of the structure detectors present in the Haar-like features, with the Dissociated Dipoles ability to use non local information. In the experiments we carried out, this new feature set obtains better results in all problems we test, compared with the use of Haar-like features and Dissociated Dipoles.

In order to test the performance of each method, and compare the different methods, we use a set of public databases, which covers face detection, text detection, pedestrian detection, and cars detection. In addition, our methods are tested to face a traffic sign detection problem, over large databases containing both, road and urban scenes.

Resum

Des dels principis de la informàtica, s'ha intentat dotar als ordinadors de la capacitat per realitzar moltes de les tasques quotidianes de les persones. Un dels problemes més estudiats i encara menys entesos actualment és la capacitat d'aprendre a partir de les nostres experiències i generalitzar els coneixements adquirits.

Una de les tasques inconscients per a les persones i que més interès està despertant en àmbit científics des del principi, és el que es coneix com a reconeixement de patrons. La creació de models del món que ens envolta, ens serveix per a reconèixer objectes del nostre entorn, predir situacions, identificar conductes, etc. Tota aquesta informació ens permet adaptar-nos i interactuar amb el nostre entorn. S'ha arribat a relacionar la capacitat d'adaptació d'un ésser al seu entorn amb la quantitat de patrons que és capaç d'identificar.

Quan parlem de reconeixement de patrons en el camp de la Visió per Computador, ens referim a la capacitat d'identificar objectes a partir de la informació continguda en una o més imatges. En aquest camp s'ha avançat molt en els últims anys, i ara ja som capaços d'obtenir resultats "útils" en entorns reals, tot i que encara estem molt lluny de tenir un sistema amb la mateixa capacitat d'abstracció i tan robust com el sistema visual humà.

En aquesta tesi, s'estudia el detector de cares de Viola i Jones, un dels mètode més estesos per resoldre la detecció d'objectes. Primerament, s'analitza la manera de descriure els objectes a partir d'informació de contrastos d'il·luminació en zones adjacents de les imatges, i posteriorment com aquesta informació és organitzada per crear estructures més complexes. Com a resultat d'aquest estudi, i comparant amb altres metodologies, s'identifiquen dos punts dèbils en el mètode de detecció de Viola i Jones. El primer fa referència a la descripció dels objectes, i la segona és una limitació de l'algorisme d'aprenentatge, que dificulta la utilització de millors descriptors.

La descripció dels objectes utilitzant les característiques de Haar, limita la informació extreta a zones connexes de l'objecte. En el cas de voler comparar zones distants, s'ha d'optar per grans mides de les característiques, que fan que els valors obtinguts depenguin més del promig de valors d'il·luminació de l'objecte, que de les zones que es volen comparar. Amb l'objectiu de poder utilitzar aquest tipus d'informacions no locals, s'intenta introduir els dipols dissociats en l'esquema de detecció d'objectes.

El problema amb el que ens trobem en voler utilitzar aquest tipus de descriptors, és que la gran cardinalitat del conjunt de característiques, fa inviable la utilització de l'Adaboost, l'algorisme utilitzat per a l'aprenentatge. El motiu és que durant el procés d'aprenentatge, es fa un anàlisi exhaustiu de tot l'espai d'hipòtesis, i al ser tant gran, el temps necessari per a l'aprenentatge esdevé prohibitiu. Per eliminar aquesta limitació, s'introdueixen mètodes evolutius dins de l'esquema de l'Adaboost i s'estudia els efectes d'aquest canvi en la capacitat d'aprenentatge. Les conclusions extretes són que no només continua essent capaç d'aprendre, sinó que la velocitat de convergència no és afectada significativament.

Aquest nou Adaboost amb estratègies evolutives obre la porta a la utilització de conjunts de característiques amb cardinalitats arbitràries, el que ens permet indagar en noves formes de descriure els nostres objectes, com per exemple utilitzant els dipols dissociats. El primer que fem és comparar la capacitat d'aprenentatge del mètode utilitzant les característiques de Haar i els dipols dissociats. Com a resultat d'aquesta comparació, el que veiem és que els dos tipus de descriptors tenen un poder de representació molt similar, i depenent del problema en que s'apliquen, uns s'adapten una mica millor que els altres. Amb l'objectiu d'aconseguir un sistema de descripció capaç d'aprofitar els punts forts tant de Haar com dels dipols, es proposa la utilització d'un nou tipus de característiques, els dipols dissociats amb pesos, els quals combinen els detectors d'estructures que fan robustes les característiques de Haar amb la capacitat d'utilitzar informació no local dels dipols dissociats. A les proves realitzades, aquest nou conjunt de característiques obté millors resultats en tots els problemes en que s'ha comparat amb les característiques de Haar i amb els dipols dissociats.

Per tal de validar la fiabilitat dels diferents mètodes, i poder fer comparatives entre ells, s'ha utilitzat un conjunt de bases de dades públiques per a diferents problemes, tals com la detecció de cares, la detecció de texts, la detecció de vianants i la detecció de cotxes. A més a més, els mètodes també s'han provat sobre una base de dades més extensa, amb la finalitat de detectar senyals de trànsit en entorns de carretera i urbans.

Mathematical notation

This section describes the mathematical notation used in this thesis. Although the mathematical formulation is limited to the minimum necessary to achieve a proper understanding of this work, some of the topics require to be defined mathematically. Notation will be used to guarantee a consistent notation for shared aspects, while each section will describe the more specific notation.

Vectors are denoted by lower case bold Roman letters such as $\mathbf{x}$, being assumed to be column vectors. When a superscript T is used, it denotes the transpose of a matrix or vector, so that $\mathbf{x}^T$ will be a row vector. Matrices are denoted using uppercase bold roman letters, such as $\mathbf{M}$. The notation $(w_1, \ldots, w_M)$ denotes a row vector with M elements, while the corresponding column vector is written as $w = (w_1, \ldots, w_M)^T$.

Notation $[a, b]$ is used to denote the closed interval from a to b, that is the interval including the values a and b themselves, while (a, b) denotes the corresponding open interval, that is the interval excluding a and b. Similarly, $[a, b)$ denotes an interval that includes a but excludes b. When the interval corresponds to a set of values instead of a range of values, it is denoted using $\{a, b, \ldots, d\}$. The $M \times M$ identity matrix is denoted $\mathbf{I}_M$, which will be abbreviated to $\mathbf{I}$ where there is no ambiguity about it dimensionality. It has elements I_{ij} that equal 1 if $i = j$ and 0 if $i \neq j$.

Notation $f : D_i \mapsto D_o$ denotes that a function f takes inputs on a domain D_i and generates outputs to the codomain D_o. To define the domain and codomain sets, blackboard bold letters $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{R}$ are used to denote naturals, integers, and reals respectively. To denote that the domain has multiple values (i.e the function takes two parameters), a cartesian product $D_1 \times \ldots \times D_N$ of parameters domains is used. In cases where the number of parameters is variable or large, parameters with the same domain are grouped using $\{D\}^N$. For instance, notation $f : \{\mathbb{R}^L\}^N \mapsto \mathbb{Z}$ denotes that a function f takes N parameters, where each parameter is a $L-$dimensional real vector, and returns an integer value. When the domain is restricted, the above notation can be used to replace the generic domains with ranges or sets of values. For instance, $f : \mathbb{R} \times (0, 1) \mapsto \{-1, 1\}$ denotes that a function f takes two input parameters, where the first parameter is an arbitrary real value and the second parameter can only be a value between 0 and 1. The output value of that function can only take values -1 and 1.

The expectation of a function $f(x, y)$ with respect to a random variable x is denoted by $\mathbb{E}_x[f(x, y)]$. In situations where there is no ambiguity as to which variable is being averaged over, this will be simplified by omitting the suffix, for instance $\mathbb{E}[x]$. If the distribution of x is conditioned on another variable z, then the corresponding conditional expectation will be written $\mathbb{E}_x[f(x)|z]$. Similarly, the variance is denoted $var[f(x)]$, and for vector variables the covariance is written $cov[\mathbf{x}, \mathbf{y}]$. We shall also use $cov[\mathbf{x}]$ as a shorthand notation for $cov[\mathbf{x}, \mathbf{x}]$.

If we have N values $x_1, \ldots, x_N$ of a D-dimensional vector $\mathbf{x} = (x_1, \ldots, x_D)^T$, we can combine the observations into a data matrix $\mathbf{X}$ in which the n^{th} row of $\mathbf{X}$ corresponds to the row vector x_n^T. Thus the n, i element of $\mathbf{X}$ corresponds to the i^{th} element of the n^{th} observation x_n. For the case of one-dimensional variables we shall denote such a matrix by x, which is a column vector whose n^{th} element is x_n. Note that $\mathbf{x}$ (which has dimensionality N) uses a different typeface to distinguish it from x (which has dimensionality D).

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

The detection and classification of objects in images that have been acquired in unconstrained environments is a challenging problem because objects can occur under different poses, lighting conditions, backgrounds and clutter. This variation in the object appearance makes unfeasible the design of handcrafted methods for object detection. Although this problem has been the subject of research from the early beginning of the computer vision field, it has not been until the recent past years that researchers have developed generic object recognition systems for a broad class of real world objects. The key point for this achievement has been the use of a machine learning framework that makes use of very large sets of sample images to learn robust models: Given a training set of n pairs $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ is the ith image and y_i is the category of the object present in $\mathbf{x}_i$, we would like to learn a model, $f(\mathbf{x}_i) = y_i$ that maps images to object categories.

State-of-the-art methods for visual recognition involve two steps. First, a set of visual features are extracted from the image and the object of interest is represented using these features. Feature selection plays a crucial role in recognition: it facilitates the identification of aspects that are shared by objects in the same class, despite their variability in appearance, and it supports discrimination between objects and between classes that can be highly similar. In the second step a classification rule is learned from the chosen feature representation in order to recognize different instances of the object. Depending on the extracted features, different classification methodologies have been proposed in the literature.

Regarding the first step, there are two main approaches to deal with the feature extraction problem (see Fig. 1.2):

- **Holistic methods** use the whole object image, that corresponds to a window of the image where the object has to be detected, to define and extract a set of features that will represent a global view of the object. These systems are typically based on defining a template matching strategy by comparing image windows to different m "templates" and recording in a vector the similarity measures. Templates can be learned from data (f.e. using Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Non Negative Matrix Factorization (NMF) or some form of artificial neural net) or can be defined *a*

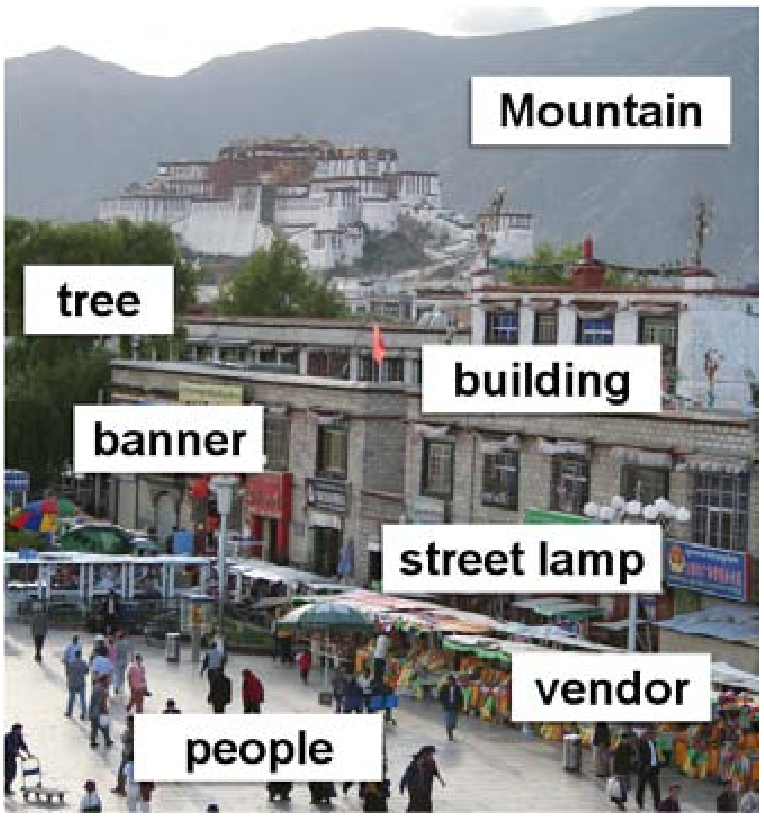


Figure 1.1: Multiple categories into a sample image [FFFT07].

priori (f.e. using a fixed wavelet dictionary or Gabor filter responses). Thus an image $\mathbf{x}_i$ can be considered to be a vector $(x_{i,1}, \ldots, x_{i,m})$ of m scalar values corresponding to m similarity measures.

- **Local methods** model an object as a collection of local visual features or "patches". Thus an image $\mathbf{x}_i$ can be considered to be a vector $(\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,m})$ of m patches. Each patch $\mathbf{x}_{i,j}$ has a feature-vector representation $F(\mathbf{x}_{i,j}) \in \Re^d$; this vector might represent various features of the appearance of a patch, as well as features of its relative location and scale. We can choose from a wide variety of features, such as the fragments-based representation approach of Ullman [US00], the gradient orientation-based SIFT [Low99], or some forms of geometric invariant descriptors.

Regarding the second step, there are two main approaches for defining the classification rule for the object representation $\mathbf{x}_i$ (see Fig. 1.3): to use a powerful m-dimensional classifier $f(\mathbf{x}_i)$ to learn a rule for assigning the object category, or to use a classification rule $F(\mathbf{x}_i)$ based on the combination of the classification results

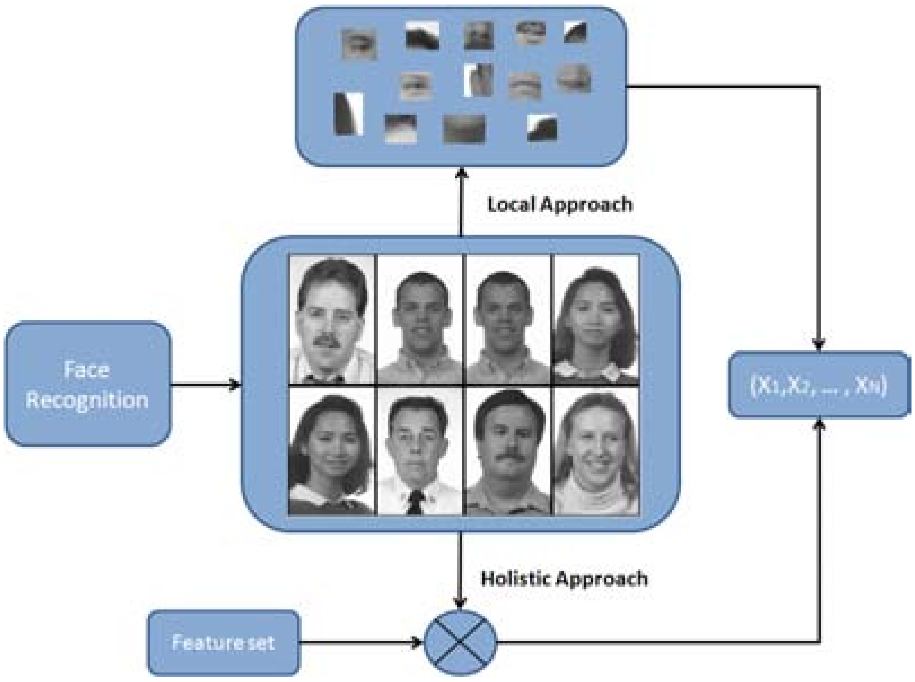


Figure 1.2: A representation of the two main approaches to image representation

of several "simple" classifiers $f_n(\mathbf{x}_i)$. In the first case we can use Nearest Neighbor classifier, Support/Relevance Vector Machines, neural networks, etc. In the second case, the most successful approaches have been based on different versions of the AdaBoost algorithm [FS96], which are based on a weighted combination of "weak" classifers $F(\mathbf{x}_i) = \sum_t \alpha_t f_t(\mathbf{x}_i)$.

In this thesis, the object detection problem is faced using the strategy proposed by Viola & Jones in the context of face detector [VJ01], one of the most extended approaches in the literature. The authors use the Adaboost algorithm in order to combine simple decision stumps based on Haar-like features in order to build a cascade of classifiers. The attractiveness of their approach relies on the combination of a widely studied and discussed learning algorithm with a robust type of features with a low computational cost and large description power. The final cascaded architecture of the detection allows a fast detection process at the same time that improves the learning capabilities. The initial schema of Viola & Jones, was extended by Lienhart et al. in [LM02], where the relationship between the used features and the learning process was studied. The classical Haar-like feature set is extended adding new configurations of regions, obtaining a larger feature set. With that new feature set, the authors compare the learning capabilities of the Adaboost algorithm using both feature sets, and conclude that larger feature sets not only obtain better results but improve the convergence speed of the Adaboost.

Results of Lienhart et al., motivates us to study other feature sets with similar

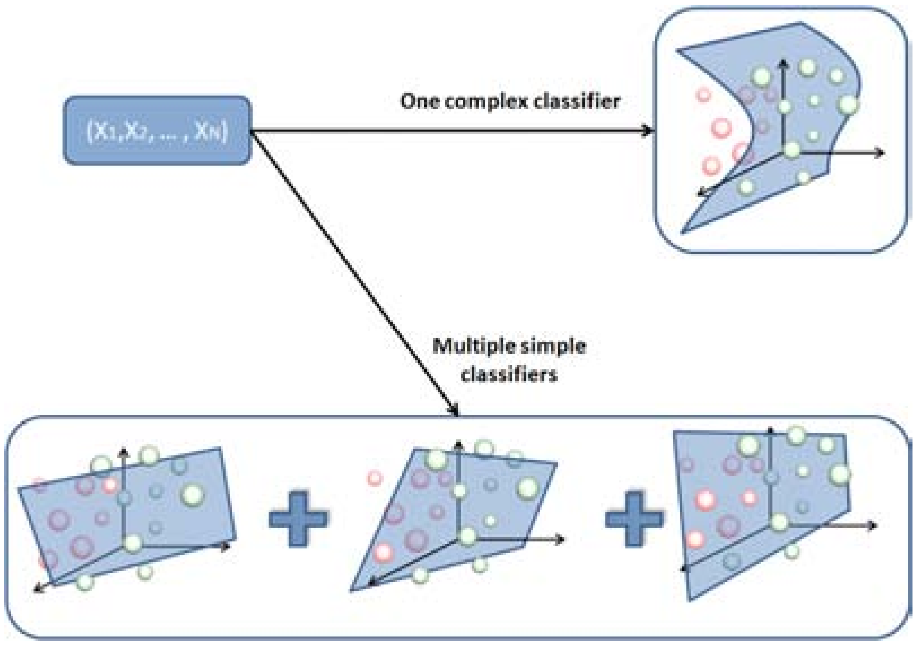


Figure 1.3: The two main classification approaches.

properties to Haar-like features. This is the case of the dissociated dipoles of Balas & Sinha [BS03], another biological inspired feature set which removes the contiguity restriction between the rectangular regions. In this case we do not have a set of contiguous regions with a predefined position and size relations, but we have two regions that can have an arbitrary position and size. Dissociated dipoles introduce non-local information of the object, because now we can compare disjoint regions of the object. Although this feature set was applied to object detection, the Viola & Jones approach was never used. The main reason is that the learning time of this approach is closely related to the cardinality of the feature set, and therefore, the use of Adaboost with large cardinality feature sets becomes unfeasible. This fact is explained because Adaboost performs an exhaustive search over all the possible weak classifiers, in other words, over all the possible combinations of feature and threshold value.

Arrived to this point, we consider a set of facts from the literature: The work of Viola & Jones [VJ01] is a widely used approach that allows real-time object detection with good performance, Lienhart et al. [LM02] suggest that larger feature sets improve the results and the convergence of the classical approach, and finally Dissociated dipoles of Balas & Sinha [BS03] is a feature set larger than the Haar-like feature and is demonstrated to be a good descriptor on object detection problems. In addition to these facts, we have a computational limitation of the learning method that makes infeasible the combination of a good object detection approach with a good feature set in order to combine the benefits of the works of Viola & Jones and Balas & Sinha in order to verify the hypothesis suggested by Lienhart et al.

The objective of this thesis is to deal with that limitation, allowing the use of large cardinality feature sets in the Adaboost scheme. Our solution relies on the reformulation of the object detection problem in terms of a function optimization problem. The result is a parametric object detection model, where the goal is to find the parameters which minimize a given error function. Although we obtain a reduced number of parameters, the cardinality of the search space avoids the use of exhaustive methods, therefore, we need to resort to optimization approaches. Moreover, some parameters are unsorted discrete values, which is an additional inconvenient to their optimization.

There are many different approaches to deal with optimization problems, most of them based on gradient descent, as line search methods, normalized steepest methods or the Newton steps method. All these methods require a differentiable function, and uses the gradient direction to move from a certain solution to a better one. In addition, most of them only support parameters which take their values on a sorted group, such as integers or reals. Apart of the fact that we have parameters which are not valid for some of these methods, it is easy to infer that the error function present a large amount of discontinuities, and therefore not differentiable in general. An alternative to the gradient based functions are the Darwin Machines, a family of approaches which emulates the natural evolution of the species defined by Darwin in order to solve optimization problems.

Given the nature of our classification function, it seems logical to assume that Darwin Machines are a good choice, and are studied and successfully used to learn object detectors for several problems. The most studied and used methods of this field are the genetic algorithms, an implementation of Darwin Machine using a chromosome based representation of the solution space, where each chromosome represents a possible solution to the problem (a set of parameters). Finally, their use mutation and crossover operators and the concept of Natural Selection to evolve a certain population of potential solutions to a better one. At practice, these methods perform an intelligent random search over the solution space.

Apart of Darwin Machines inspired on genetics, from the publication of Population Based Incremental Learning (PBIL) in 1995 by Baluja and Caruana [BC95], a new family of methods is striving to find a place in this world. These methods, which are referred as Probability Darwin Machines (PDM), use an evolutionary strategy to learn a probability model of the good individuals, and thus, of the best solutions. In practice, what is done in the PDM is to summarize the knowledge contained in a set of individuals, into a probability model. Instead of mutations and cross-overs, in the PDM framework, the probability model is sampled in order to generate the new population of potential solutions.

The use of an evolutive approach allows to take advantage of the boosting strategy in large cardinality feature spaces, such as the dissociated dipoles. Once the limitation of the classical approach is broken, we compare the descriptive power of Haar-like features with the dissociated dipoles one. The results show that the cardinality is important, but is not the only requirement to improve the Adaboost performance. Some of the patterns in the Haar-like features cannot be simulated by the dissociated dipoles, and add an extra description power that compensates the lower cardinality of the feature set. At the end both feature sets get similar performance. This result motivates the definition of a new feature set that shares the benefits of the dissociated dipoles (non local information) and Haar-like features (structure detection patterns). This goal is accomplished with the weighted dissociated dipoles, which obtained better results than dissociated dipoles and Haar-like features in all of the tested problems.

Since this thesis combines object detection and evolutionary computation methodologies, it is structured in three main parts:

Firstly, we introduce the object detection problem from a theoretical point of view. After a general view of the problem and its context, we analyze the classical learning approach with different feature sets.

The second part is an introduction to evolutionary computation and the most applied algorithms, with special attention to PDM. Since that last methods are based on probabilistic models, some of the most used models are discussed.

Finally, once the concepts and methodologies of both fields are defined, the third part is focused on the combination of both fields. At the beginning of this part we formulate the object detection problem in terms of a function optimization problem, and then apply evolutionary approaches in order to solve it.

Although the goal of this thesis is to provide a new object detection learning approach, and methods and discussions contained in this thesis are applicable to any object detection problem, during the thesis we take special attention to a the mobile mapping problem, where a mobile vehicle is used in order to collect different types of cartographic data. The collaboration with the *Institut Cartogràfic de Catalunya*, where a new mobile mapping system has been developed, allowed us to test our methods over large datasets of real world data, either in road scenes and urban scenes. In this context, we face traffic signs detection problem, text detection problem, and we are studying the use of our methods to detect other urban objects. In addition, this collaboration makes possible a better understanding of the practical issues that rarely are find working with common databases, which are prepared to test methods. Moreover, it give us an external point of view for the methodologies and results, giving a useful feedback which allowed to enrich our work.

Apart of some appendixes extending parts of the thesis, such as the used image databases, and the basis of the statistical analysis we use to evaluate our results, the thesis is complemented by one appendix where the traffic signs problem is addressed, and another appendix with a previous approach to address the problem of large cardinality feature sets.

Chapter 2

Object detection

2.1 Rare event detection

From the beginnings, most of the *Information Technologies* applications have addressed tedious or difficult problems for the humans, commonly based on repeated actions or complex calculation tasks. The applications that the *Information Technologies* wants to solve when applied to computer vision, and more concisely the object detection problem, are tasks that belong to the natural behavior of humans, and therefore, sometimes it is difficult to view the complexity of the problem. This chapter explores the object detection as an instance of object recognition, defining the main concepts and highlighting the difficulties of building an artificial system that emulates de human visual system behavior.

2.1.1 Object recognition

Object recognition is one of the most important, yet least understood, aspects of visual perception. For many biological vision systems, the recognition and classification of objects is a spontaneous, natural activity. Young children can recognize immediately and effortlessly a large variety of objects [Ull96].

In contrast, the recognition of common objects is still way beyond the capabilities of artificial systems, or any recognition model proposed so far. The brain generalizes spontaneously from visual examples without the need for explicit rules and laborious instruction, being able to recognize these objects under a huge variety of viewing conditions. On the contrary, computers must be programmed to recognize specific objects with fixed, well defined shapes. It is considerably more difficult to capture in a object recognition system the essence of a dog, a house or a tree, which is the kind of classification that is natural and immediate for the human visual system.

Is not easy to define the term *object recognition* in a simple, precise and uncontroversial manner. What do we mean, exactly, when we say that we *recognize an object*? The simplest answer might be: *"naming an object in sight."* This answer is not entirely unambiguous, because in recognizing an object, we wish sometimes to identify an individual object or a specific *token* (such as *my car*), while in other cases recognition means identifying the object as a member of a certain class, or a type (*a truck*). We will name **identification** the first case, were we want to identify an individual object, and **classification** the second one, were we only want to know the pertinence to a certain class.

Furthermore, an object may belong to a number of classes or categories simultaneously (eg. my cat, a Siamese cat, a cat, an animal). In the same way, in an image we can find multiple objects, and each one can be composed of differentiated parts. The finality of the recognition system will define the level of classification we require.

Object recognition seems an easy problem that could be overcome by using a sufficiently large and efficient memory system. When performing recognition, we are trying to determine whether the image we currently see corresponds to an object we have seen in the past. It might be possible, therefore, to approach object recognition by storing a sufficient number of different views associated with each object, and then comparing the image of the currently viewed object with all the views stored in memory [AMP87]. Several mechanisms, known as associative memories, have been proposed for implementing this *direct* approach to recognition. A major problem with this approach is that it relies on a simple and restricted notion of similarity to measure the distance between the input image and each of the images stored previously in memory. As is showed in [Ull96], the use of a simple image comparison is insufficient by itself to cope with the large variations between different images of a given object. Summarizing, for the general problem of visual object recognition this direct approach is insufficient for two reasons: The first, the space of all possible views of all the objects to be recognized is likely to be prohibitively large, and the second, and more fundamental reason is that the image to be recognized will often not be sufficiently similar to any image seen in the past. The differences in this second case can be produced by several reasons: Viewing position, photometric effects, object setting or changing shape.

- *Viewing position:* Three-dimensional objects can be viewed from a variety of viewing positions (directions and distances), and these different views can give rise to widely different images (see Fig. 2.1).

- *Photometric effects:* These include the positions and distribution of light sources in the scene, their wavelengths, the effects of mutual illumination by others objects, and the distribution of shadows and specularities (see Fig. 2.2).

- *Object setting:* In natural scenes, objects are rarely seen in isolation: they are usually seen against some background, next to, or partially occluded by other objects. Some examples are shown in Fig. 2.3:

- *Changing shape:* Many objects, such as the human body, can maintain their identity while changing their $3-$D shape. Changing objects, such as a pair of scissors, can sometimes be composed of rigid sub-parts. Other objects may undergo non-rigid distortions, for example, faces undergoing complex transformations due to facial expressions or hands (see Fig. 2.4).

A large variety of methods have been proposed for the task of visual object recognition, some of them as models of human vision, others as possible schemes for machine

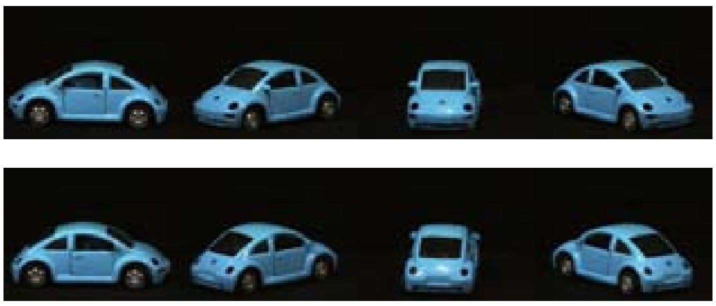


Figure 2.1: Different views of a car. Images are taken from from [GBS05]
.

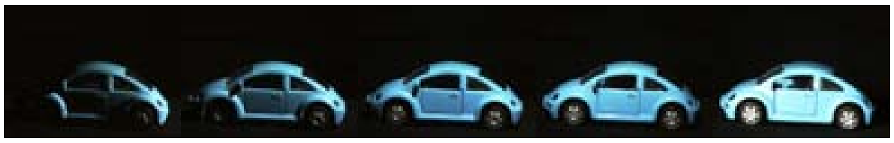
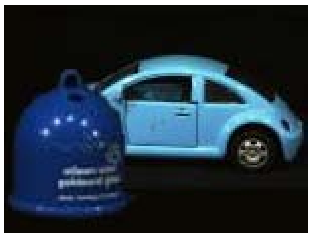
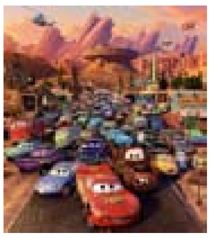


Figure 2.2: Different illumination conditions for an object. Images are taken from from [GBS05]
.



(a) Occlusion



(b) Complex backgrounds [DP06]

Figure 2.3: Different settings.

vision. Some of the proposed schemes are general in nature, others were developed for specific application domains (see [Bin81, BJ85] for reviews). In [Ull96], Ullman classifies the basic approaches to recognition in three main classes, based on their approach to the regularity problem:

- *Invariant properties methods:* Theories in this class assume that certain simple properties remain invariant under the transformations that an object is allowed to make.
- *Parts decomposition methods:* Theories in this class relies on the decomposition of objects into parts. This leads into the notations of symbolic structural

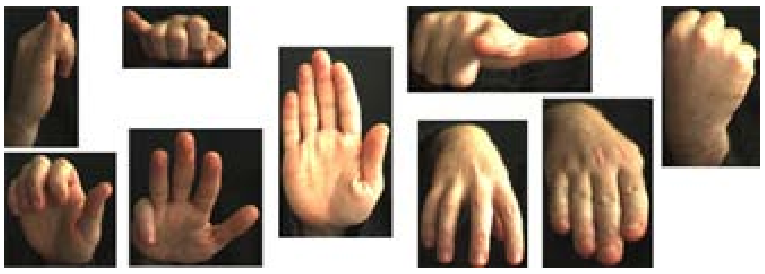


Figure 2.4: Example of a changing shape object [DBS05]

.

descriptions, feature hierarchies and syntactic pattern recognition.

- *Alignment methods:* The main idea for theories in this class is to compensate for the transformations separating the viewed object and the corresponding stored model, and then compare them.

This classification is a taxonomy of the underlying ideas, not of existing schemes. That is, a given scheme is not required to belong strictly to one of these classes, but may employ one or more of these ideas. A successful recognition scheme may in fact benefit from incorporating key ideas from all three classes.

2.1.2 Object detection

Although theoretical object recognition defines the general framework for the problem faced in thesis, we need to introduce more accurate and practical definitions to the specific case of study, the object detection problem. We use the term *object detection* to describe a specific kind of *classification*, where we only have two classes: the class *object* and the class *no object*.

Given an input image, the goal of object detection is to return the instances of the class *object* in this image. Therefore, object detection is often posed as a search and classification problem: a search strategy generates potential image regions and a classifier determines whether or not they are an object. The standard approach is brute-force search, in which the image is scanned in raster order using a $n \times m$ pixels window over multiple image scales. For each scale and position the window is classified. When the brute-force search strategy is used, object detection is a **rare event detection** problem, in the sense that among the millions of image regions, only few of them are objects.

When facing **rare event detection** problems, we are restricted to use a fast classification strategy in order to discard those millions of windows that do not contain any object. This fact must be considered on the three main aspects of the classifier: How objects are described, how objects are modeled, and how image regions are analyzed.

Although this thesis is based on the **rare event detection** strategy, in the following, the **bag of words**, another widely used strategy is analyzed in order to give a more general vision of the problem. Finally, the rest of the sections of this chapter exposes how to face the different aspects of a **rare event detection** strategy.

2.2 Bag of Words object detection

A simple approach to classifying images is to treat them as a collection of regions, describing only their appearance and ignoring their spatial structure. Similar models have been successfully used in the text community for analyzing documents and are known as *bag of words* models, since each document is represented by a distribution over fixed vocabulary(s). This type of representation has been widely used in document classification, such as thematic classification or spam detection. A part of those commonly supervised learning tasks, using this representation, methods such as probabilistic latent semantic analysis (pLSA) [Hof99] and latent Dirichlet allocation (LDA) [BNJ03] are able to extract coherent topics within document collections in an unsupervised manner.

This methodology has been imported to the computer vision field, using the analogous definition. When *Bag of Words* is applied over images, the image is considered the document, and the words are fragments of this image. As in the case of document analysis, this approach has been widely used in object recognition and segmentation. Recently, Fei-Fei et al. [FFP05] and Sivic et al. [SRE$^+$05] have applied the pLSA and LDA, extending the coherent topics extraction to the visual domain. Although when this approach is applied over images some authors call them *bag of features*, in this thesis we use the original name.

Working with *Bag of Words* approach, one can find four common processes:

Patch extraction: Each image in the samples set is divided in a generally large set of small patches.

Patches description: The patches are described as a vector. This process can be done using a set of features, where the $i-$th position of the vector corresponds to the value of the $i-$th feature, or using a global method which converts an image to a vector. We put this process here because most of the classical approaches to *Bag of Words* use a description method [FFP05, VS04, SRE$^+$05], nevertheless, in some other works the patches are used as images, and therefore, no description method is applied [UVNS02, US00, USVN01, BU01].

Dictionary building: Once all the patches has been described, the next process is to build a set of representative patches. The underlying idea is to reduce the number of patches to describe objects with. The result is a smaller number of representative patch descriptors (words).

Objects representation: Once the dictionary is built, any object can be represented by means of the apparition of words. Given an object image, the patches are extracted and described. Finally, each patch votes to the most similar word in the dictionary.

Although there is a wide variety of methods to perform each task, only the most representative are considered in the next section. Each process is described, introducing some examples of methodologies.

2.2.1 Patch extraction

The decomposition of an object into a set of patches can be addressed using different approaches:

Regular grid: The input image is divided in a fixed number of cells, using overlapping or not (see Fig. 2.5), and each one of these cells are used as a patch. Examples of this method are found in [VS04] and [FFP05].

Interest point: One of the most nowadays used approaches to patches extraction is the one based on points of interest. The undergoing idea is to identify regions in the input image that shares some interesting property, such as stability throw different scales and repeatability (see Fig. 2.6). Examples of this approach are found in [Low99], [DWF$^+$04], [FFP05], and [SRE$^+$05].

Invariant regions : A good property for a patch is the invariance to different transformations. This property usually is required to the descriptor, but some works introduce some normalization stages to the patch generation step. This is the case of Mikolajczyk and Schmid [MTGM04], where affine invariant interest points are extracted. This approach deal with significant affine transformations including large scale changes, avoiding significant changes in the point location as well as in the scale and the shape of the neighborhood of an interest point.

Other methods: Although most works use one of the previous methods, in the literature we can find other works that uses other methods in order to find the patches. This is the case of [VNU03] where a random sampling strategy is used or in [BDF$^+$03], where the authors use patches resulting from a segmentation process.

Although we can found applications of all these methods, the comparison between the *regular grid* approach and the use of an *interest point* detector made by Fei-Fei & Perona [FFP05], concluded that better results are obtained using a simple *regular grid* division of the image.

2.2.2 Patch description

Once the patches have been created using one of the above methods, each patch is described using a numerical vector which represents any characteristic of the patch, that is, given a patch P of the image, we define a function $\mathcal{F} : P \rightarrow \mathbb{R}^D$, where D is the length of the descriptor and corresponds to the dimension of the output space. An even larger variety of feature descriptors has been proposed, like Gaussian derivatives [FRKV94], moment invariants [MTGM04], complex features [Bau00, SZ02], steerable filters [FA91], phase-based local features [CJ03a], and descriptors representing the distribution of smaller-scale features within the interest point neighborhood. In the following, some of the most relevant descriptors are introduced and compared:

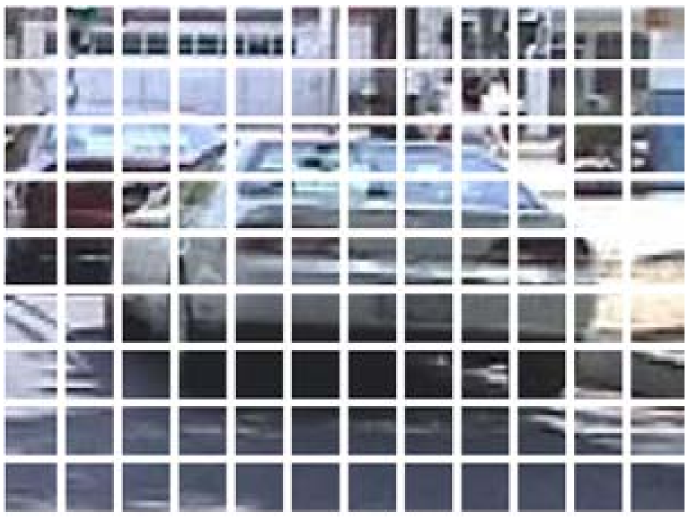


Figure 2.5: Patch extraction based on a fixed grid.

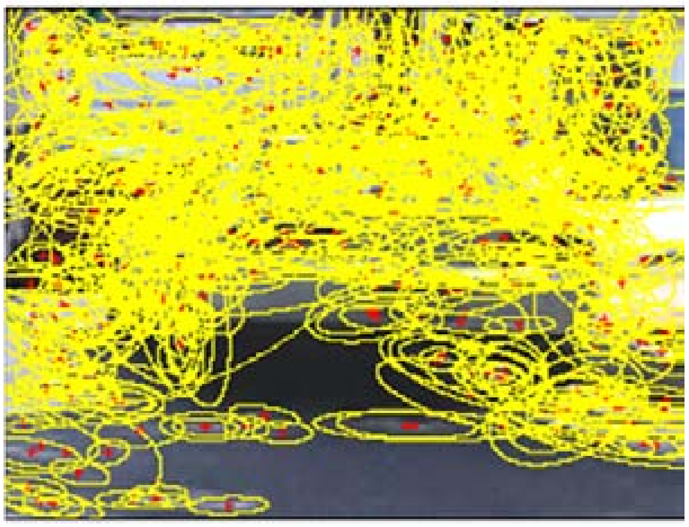


Figure 2.6: Patch extraction based on points of interest.

SIFT

The Scale Invariant Feature Transform (SIFT) introduced by Lowe in [Low04], is one of the most successful descriptors in the *bag of words* approach. Its mixing of crudely localized information and the distribution of gradient related features seems to yield good distinctive power while fending off the effects of localization errors in terms of scale or space. Using relative strengths and orientations of gradients reduces the effect of photometric changes.

The SIFT descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the center of the patch, as shown on Fig. 2.7 left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4×4 subregions, as shown on Fig. 2.7 right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region.

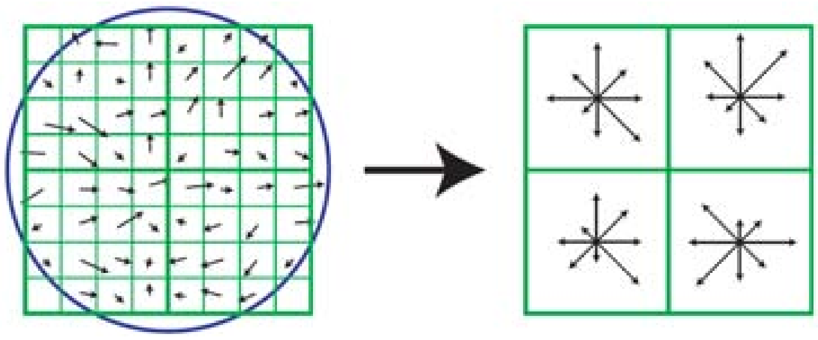


Figure 2.7: SIFT descriptor schemata [Low04]. On the left, it is shown the sampling process of the path, and the weighting Gaussian window. At right, the resulting gradient magnitudes for each region.

In order to obtain rotation invariance, the orientations are computed referred to the predominant direction, which is obtained as follows:

The Gaussian-smoothed image $L(x, y, \sigma)$ at a certain keypoint (x, y) and scale σ is taken so that all computations are performed in a scale-invariant manner. The gradient magnitude, $m(x, y)$, and orientation, $\theta(x, y)$, are computed using pixel differences:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{2.1}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \tag{2.2}$$

For every pixel in a neighboring region around a keypoint in the Gaussian-blurred image L, the magnitude and direction calculations for the gradient are computed. Those values are stored in an orientation histogram with 36 bins, with each bin covering 10 degrees, weighting each sample in the neighboring window by its gradient magnitude and by a Gaussian-weighted circular window with a σ of 1.5 times that of the scale of the keypoint. The peaks in this histogram correspond to dominant orientations. Once the histogram is filled, the orientations corresponding to the highest peak and local peaks that are within 80% of the highest peaks are assigned to the keypoint. In the case of multiple orientations being assigned, an additional keypoint is created having the same location and scale as the original keypoint for each additional orientation.

Although the length of the output descriptor depends on how the region is sampled, the most common configuration assigns a vector of 128 values to the path. In [MS05], Mikolajckyz et al. stated that this descriptor outperform most of the state-of-the-art descriptors.

SURF

The *SURF* descriptor introduced by Bay et al. [BTG06], is based on the similar principles of Lowe's *SIFT* descriptor [Low04], with a complexity stripped down even further. As in the case of *SIFT*, a first alignment step is performed in order to obtain rotation invariance. In addition, the authors introduce a faster descriptor, the Upright-SURF *(U-SURF)* which do not perform the rotation normalization, and therefore, which is sensitive to rotation effects.

In order to extract the descriptor, the patch is split up regularly into smaller 4×4 square sub-regions. This keeps important spatial information in. For each sub-region, a few simple features are computed at 5×5 regularly spaced sample points. In contrast to the *SIFT* descriptor, in this case, the gradient is approximated using simple Haar wavelets. For reasons of simplicity, we call d_x the Haar wavelet (see Sec. 2.3.1) response in horizontal direction and d_y the Haar wavelet response in vertical direction ("Horizontal" and "vertical" is defined in relation to the predominant orientation of the normalization step). To increase the robustness towards geometric deformations and localization errors, the responses d_x and d_y are first weighted with a Gaussian centered at the center of the patch.

The wavelet responses d_x and d_y are summed up over each subregion and form a first set of entries to the feature vector. In order to bring in information about the polarity of the intensity changes, the absolute values of the responses $|d_x|$ and $|d_y|$ is calculated. Hence, each sub-region has a four-dimensional descriptor vector v for its underlying intensity structure $v = (d_x, d_y, |d_x|, |d_y|)$. The final descriptor is a vector of length 64. The properties of the descriptor for three distinctively different image intensity patterns within a subregion is shown in Fig. 2.8. A part from the *SURF*

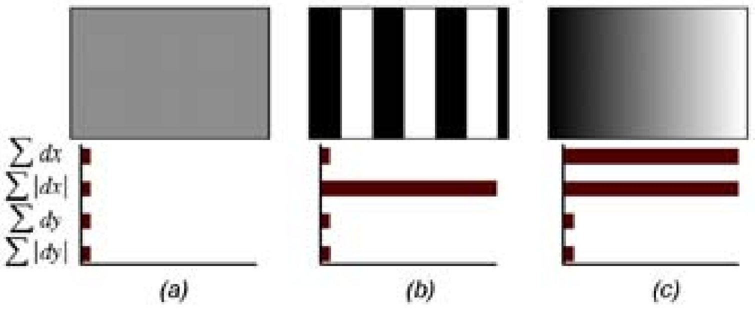


Figure 2.8: The SURF descriptor entries of a sub-region represent the nature of the underlying intensity pattern. *(a)* In case of a homogeneous region, all values are relatively low. *(b)*In presence of frequencies in x direction, the value of $\sum |dx|$ is high, but all others remain low. *(c)* If the intensity is gradually increasing in x direction, both values $\sum dx$ and $\sum |dx|$ are high [BTG06].

using a descriptor with 64 values, the authors introduce a more accurate descriptor with 128 values. It again uses the same sums as before, but now splits these values in different groups: The sums of d_x and $|d_x|$ are computed separately for $d_y < 0$ and $d_y \geq 0$. Similarly, the sums of d_y and $|d_y|$ are split up according to the sign of d_x. The comparisons made by the authors in [BTG06] concluded that both implementations of the *SURF*outperform *SIFT* results.

Gabor Jets

Another classical approach to describe images are the Gabor filters (see Sec. 2.3.2). *Gabor Jets* are the outputs (i.e., signal power) from this set of Gabor filters. In this case the length of the descriptor depends on the number of applied filters. Examples of this representation can be found in [SSCG06] for face identification or in [DC99] for facial expressions recognition.

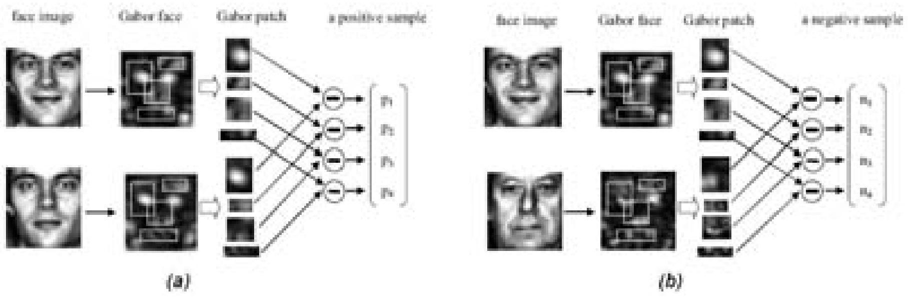


Figure 2.9: Example of a patch description and selection used in [SSCG06] for face identification. *(a)* The image is codified using an extended Gabor Jet vector. Using the distances between de different Gabor Jets, the most useful patches to represent the intra-class variability are selected for the final set of positive values.*(b)* The process is repeated, but in this case, the Gabor Jet descriptor is used to select the discriminant patches between classes.

Binary features

Apart of the methods which try to capt the relevant information using complex information, other works use simple representation methods based on binary images, commonly based on some type of edge or corner detector. In [Fle04], the boolean features they use are crude edge detectors, invariant to changes in illumination and to small deformations of the image (see Fig. 2.10).

Another example of binary representation can be found in [SBC08], where patches are represented using edge (they test different edge extraction methods, such as Canny [Can86]). In addition to the edges, the representation of the patch is complemented using the position of the object centroid (see Fig. 2.11).

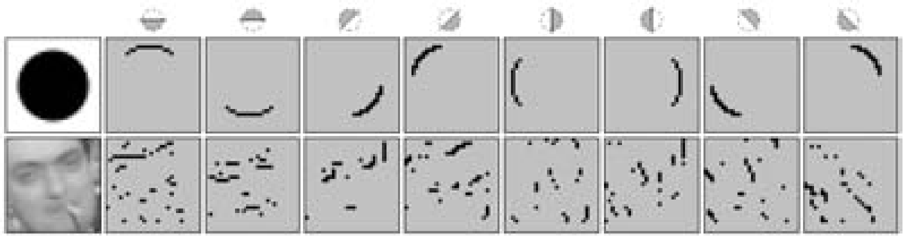


Figure 2.10: The original gray-scale pictures are shown on the left. The eight binary maps on the right show the responses of the edge detectors at every locations in the 28×28 frame, for every one of the 8 possible directions and polarities. The binary features are disjunctions of such edge detectors in small neighborhoods, which ensure their robustness to image deformations [Fle04].

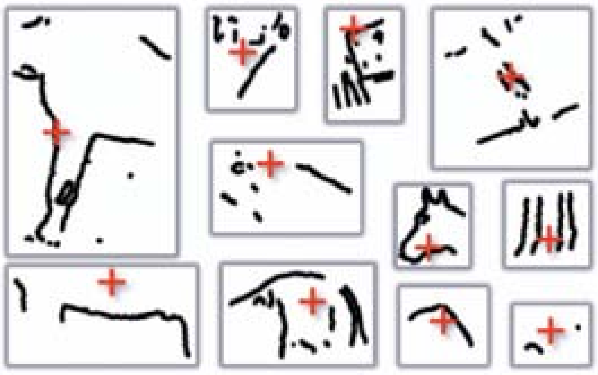


Figure 2.11: Examples of contour fragments extracted at random from the edge maps of horse images. The +s represent the fragment origins [SBC08].

2.2.3 Dictionary building

After previous processes, our training images have been converted into a large set of smaller regions. Therefore, the amount of available data can be a problem in order to perform future tasks. A classical way to solve this problem consists on building a dictionary or code-book in order to reduce the number of possible patches. This process is often performed using a clustering step, and storing the centers of the clusters as the words of the dictionary. One of the most used algorithms in order to perform this task is the K-means (see Fig. 2.12). Once the dictionary is created using the K most representative words, each patch is described using the description of their correspondent word in the dictionary. In order to find the correspondent word, a K nearest neighbors algorithm can be used.

Recently, the clustering approach has been questioned because of their strictness. The centers are extracted once and cannot be updated. In [Per08], Perronnin proposes an alternative representation for the dictionary, based on a Gaussian Mixture Models. This approach is more flexible in order to be updated with new words, even if the initial data is not available. In addition to the dictionary model, Perronnin proposes

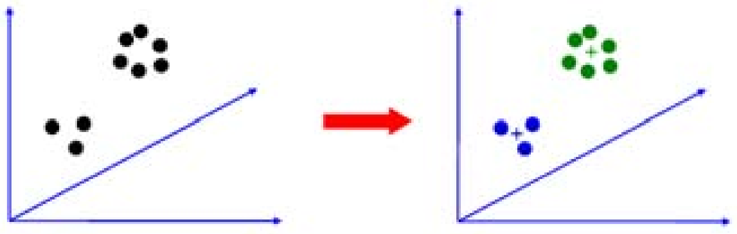


Figure 2.12: K-means clustering approach for dictionary building.

the use of two dictionaries, the universal dictionary for all the objects, and the class-specific dictionary. This extra information allows a better representation of objects.

2.2.4 Object recognition approaches

Either if a dictionary is created or the patches are used as is, the final purpose of all methods is to detect instances of the objects of interest in a given image. Although we can find a wide variety of works using different methodologies and representations for the objects, in all of them, objects need to be represented using their patches. The classical approach is based on a bottom-up process, where during the learning process, each object category is modelled using the distribution of the words (see Fig. 2.13).

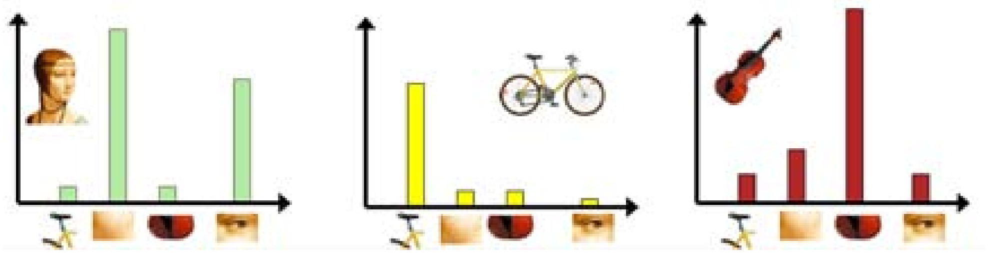


Figure 2.13: Words distribution for different categories [FFFT07].

Given an input image, the patches are extracted using the same methodology than in the learning step, and each patch is represented with their correspondent word in the dictionary. Then the model is applied in order to determine if the image contains an object or not. In contrast with the *rare event detection* approach, since in this case we do not have a window, the only information available on most of the *bag of words* approaches is the existence or absence of the object, not their location in the image. An exception can be found in the work of Lowe [Low99], where they align the interest points with the ones in the learning image, obtaining the scale and position of the object in the detection process.

2.2.5 Object localization approaches

If the precise location of the object in the image is necessary, additional tasks must be performed. Once we have an image where we know that a certain object is present, we can apply a top-down process to get the exact position and scale.

In order to find the object in the image, the class-specific words are searched in the input image. This process is commonly performed by means of some type of correlation or distance between the patch and the image. Once all the patches has been positioned, we got an approximation to the bounding box of the object.

Alternatives to this process require to store additional information with patches, as the displacement from the centroid of the object [MTEF06, SBC08]. This information is used in order to perform a voting system, where the centroid of the object is precisely located.

Finally, we can find methods as the one of Borenstein and Ullman [BU01], where each patch have a related binary mask, indicating whereas the pixel corresponds to the object or to the background. This information is used in order to perform a segmentation of the object, obtaining not only the position, but their shape.

2.2.6 Conclusions

Alter analyzing the *Bag of Words* approach, our conclusions are that in general those methods can obtain as good results as in the case of the *rare event detection* approach, but in most cases, if the generalization capability is good, the computational cost is large, and when fast methods are used (e.g. SIFT or SURF), the generalization capability decreases. In this second case, those methods are most powerful than the *rare event detection* for identification tasks, where we want to detect on object, not a family of objects.

When the *Bag of Words* approach is used without a description of the patches, the use of correlation suffers from the existence of large background regions in the patches, as is stated in [BU01]. This problem is smoothed in this work using statistical methods in order to find the part of each patch that correspond to the object and the one that corresponds to background. After that process, the authors associate a mask image to each patch in order to use only the object region of the patch on the correlation process.

In the rest of the thesis, our work is concentrated on the *rare event detection* approach. As in this case the computational complexity of the methods is a restriction, the descriptors presented on this section are not used, and more simple features are introduced in order to describe the objects. Although is not common, the features we introduce in the next section could be applied in the description of the patches. The main reason to do not use them in patch description, is because they need additional learning methods, and in the case of patches, it can suppose to increase drastically the description process time.

2.3 Features

There is no universal or exact definition of what constitutes a feature, and the exact definition often depends on the problem or the type of application. In object recognition and pattern recognition in general, the objects or phenomenons being observed must be described in a computationally plausible representation. In this context, features are the individual measurable heuristic properties of the object or phenomena of interest, and are restricted by the available perception of the world, such as sensors or transaction databases. Building proper representations has become an important issue in pattern recognition [DRdR02].

In the case of computer vision, the starting point is always a matrix with numerical values which are interpreted as an image. Depending on the application, this image can represent distances from a sensor, the spectrogram of a sound or more frequently the intensity value on a light sensor matrix. Although that numeric matrix is in fact a set of features that represent the object, in general, that initial representation of the object is not enough to perform classification tasks, and more complex features must be computed. We can define a feature as any function $f : M \mapsto \mathbb{R}$ that takes an image M as input and returns a real value as output. In general, we never use a feature alone, instead, we use groups or families of features which are referred as a feature set. While different areas of pattern recognition obviously have different features, once the features are decided, they are classified by a much smaller set of algorithms. These include nearest neighbor classification in multiple dimensions, neural networks or statistical techniques.

The description of an object by means of a set of features is known as feature extraction. Although the nature of our objects and application will suggest the features we can use, in the literature there are different levels of features that are used for different purposes. On 1969, Levine [Lev69] classify the features in two main groups: Microanalysis and macroanalysis. The main idea is that previous to the description of an image, we need to identify the most informative regions of that image, using a set of generalist features as edges, lines, etc... This initial process is called the microanalysis, while the description of the informative regions using a more problem-dependant features is called the macroanalysis. A part from nomenclature changes (i.e. microanalysis is commonly referred as feature detection), that division is still accepted. It is important to state that this is a fuzzy division, and some features can be classified in any of two groups depending on the application.

It seems clear, both from biological and computational evidence, that some form of data compression occurs at a very early stage in image processing. Moreover, there is much physiological evidence suggesting that one form of this compression involves finding edges and other features with a high information of images. Edges often occur at points where there is a large variation in the luminance values in an image, and consequently they often indicate the edges, or occluding boundaries, of the objects in a scene. However, large luminance changes can also correspond to surface markings on objects. Points of tangent discontinuity in the luminance signal (rather than simple discontinuity) can also signal an object boundary in the scene.

So the first problem encountered with modeling this biological process is that of defining, precisely, what an edge might be. The usual approach is to simply define edges as step discontinuities in the image signal. The method of localizing these discontinuities often then becomes one of finding local maxima in the derivative of the signal, or zero-crossings in the second derivative of the signal. This idea was first suggested to the AI community, both biologically and computationally, by Marr [Mar82], and later developed by Marr and Hildreth [MH80], Canny [Can86], and many others [Der87, Fle92].

Feature detection usually refers to the computation of points of interest where we compute the features. An interest point is a point in the image where the local image structure around it is rich in terms of local information contents, such that the use of interest points simplify further processing in the vision system. In general, it is desirable to be stable under local and global perturbations in the image domain, including deformations as those arising from perspective transformations (sometimes reduced to affine transformations, scale changes, rotations and/or translations) as well as illumination/brightness variations, such that the interest points can be reliably computed with high degree of reproducibility.

Once the informative regions of an object are selected, there are a large list of possible descriptors to characterize them. Most of them are based in illumination changes or on gradient information. That stage is the commonly called feature extraction. As we will state later, feature detection is sometimes mixed within the feature extraction process, and we cannot differentiate both processes. This is the case of the features used in this thesis, which are presented in the following.

2.3.1 Haar-like features

One of the most successfully used *a priori* image feature, at least for a broad class of visual objects, is known as Haar-like feature. These features, which are related to the discrete wavelet decomposition (DWT), were originally proposed in the framework of object detection by Viola and Jones [VJ01] in their face detection approach.

The foundations of the DWT go back to 1976 with the works of Croiser et al. [CEG76] and Crochiere et al. [CWF76], where a technique was devised in order to decompose discrete time signals. They named their analysis scheme as sub-band coding, which was later improved by Vetterli and Le Gall [VL89] removing the existing redundancy in the pyramidal coding scheme.

Wavelets are composed of two bases, a scaling function or *carrier* and a wavelet basis or *envelop*. The scaling basis deals with how the wavelet represents a signal over a given frequency band while the wavelet basis shows how the wavelet sees the transients in a signal on a given frequency band. The Haar basis is perhaps the simplest example of a DWT basis. Haar mother scaling function $\theta(t)$ is defined by equation 2.3 and Fig. 2.14.

$$\phi(t) = \begin{cases} 1 & \text{if } 0 \le t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

From the mother scaling function, a family of shifted and stretched scaling func-

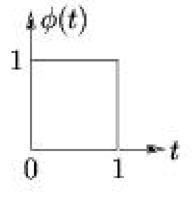


Figure 2.14: Haar wavelet mother scaling function.

tions $\{\phi_{k,n}(t)\}$ are defined by equation 2.4 and Fig. 2.15.

$$\begin{aligned} \phi_{k,n}(t) &= \forall k,n | k \in \mathbb{Z}, n \in \mathbb{Z} : \left(2^{-\frac{k}{2}}\phi(2^{-k}t-n)\right) \\ &= 2^{-\frac{k}{2}}\phi\left(\frac{1}{2^k}(t-n2^k)\right) \end{aligned} \tag{2.4}$$

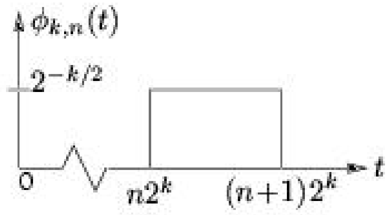


Figure 2.15: Family of shifted and stretched Haar wavelet scaling functions.

Changing the values of n and k, we can obtain different instances of the scaling function family. Observe from Fig. 2.16 that $\{\phi_{k,n}(t)|n \in \mathbb{Z}\}$ is orthonormal for each value of k parameter (along rows).

The Haar wavelet basis functions are scaled and translated versions of the *mother wavelet* $\psi(t)$ defined by equation 2.5 and Fig. 2.17.

$$\psi(t) = \begin{cases} 1 & \text{if } 0 \leq t < \frac{T}{2} \\ -1 & \text{if } 0 \leq \frac{T}{2} < t \end{cases} \tag{2.5}$$

Basis functions $\{\psi_{j,k}(t)\}$ are indexed by a *scale* j and *shift* k. Using equations 2.3 and 2.5, $\forall t, 0 \leq t < T : (\phi(t) = 1)$ the basis functions family are defined as $\{\phi(t), 2^{\frac{j}{2}}\psi(2^j t - k)|j \in \mathbb{Z} \wedge k = 0, 1, 2, ..., 2^j - 1\}$. In Fig. 2.18 the scale and shift effects are shown and finally in Fig. 2.19 some examples of Haar wavelets are shown.

The two-dimensional Haar decomposition of a square image with n^2 pixels consists of n^2 wavelet coefficients, each of which corresponds to a distinct Haar wavelet. The first such wavelet is the mean pixel intensity value of the whole image and the rest of the wavelets are computed as the difference in mean intensity values of horizontally, vertically, or diagonally adjacent squares. In Fig. 2.20, a sample of a 2D decomposition

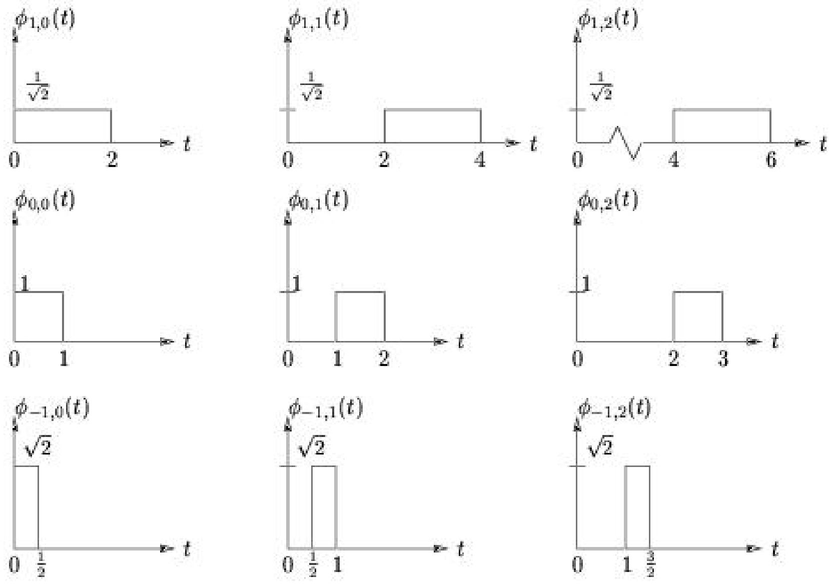


Figure 2.16: Family of Haar wavelet functions for different values of n and k.

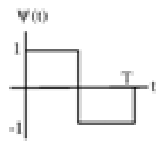


Figure 2.17: Haar *mother wavelet* function.

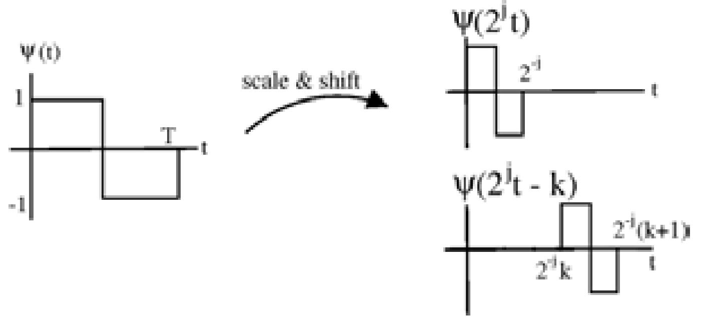


Figure 2.18: Effects of *scaling* and *shifting.*

of Lena's image is presented. Note that Haar wavelets are sensitive to edges in the image.

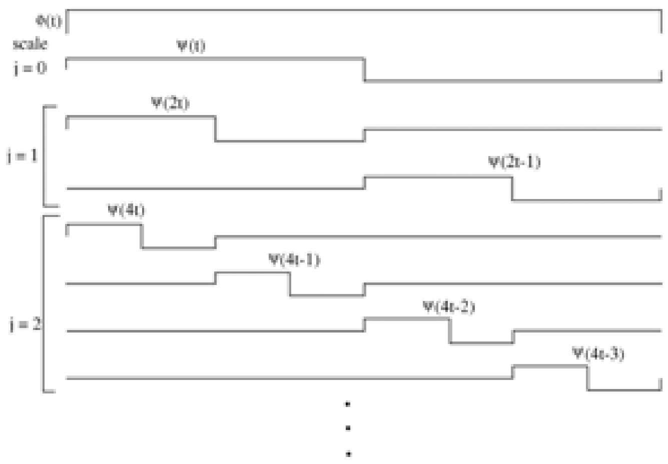


Figure 2.19: Haar wavelet samples.

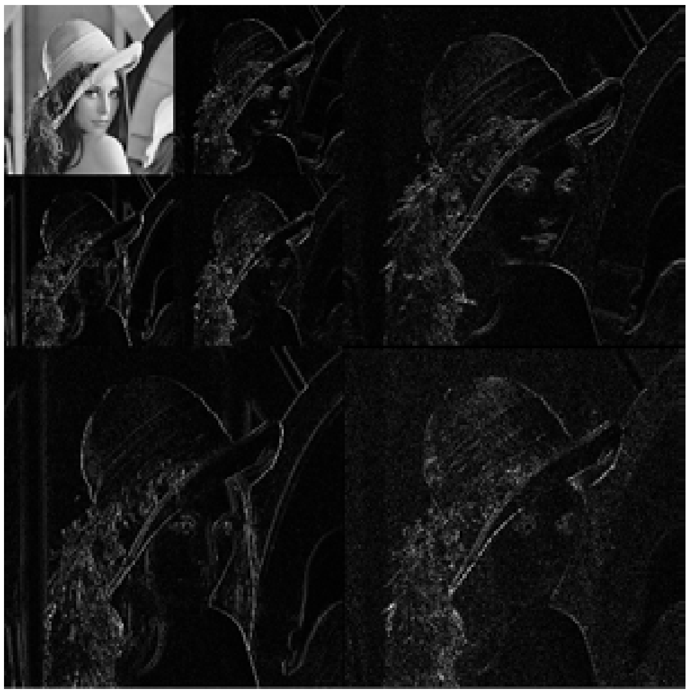


Figure 2.20: Second level decomposition of Lena Image using Haar Wavelet.

Although Haar wavelet is widely used in the image processing and image compression fields, on patter recognition is often used a simplified version so-called Haar-like features. From its first apparition in the work of Viola & Jones [VJ01] and the posterior extension in [LM02], Haar-like features has become a standard image feature when detection of a broad class of visual objects is faced. These features are based on a set of predefined patterns (see Fig. 2.21), which are defined in order to detect concrete structures in an image. Haar-like features are an extension of Haar wavelet definition to all possible adjacent rectangles in the image.

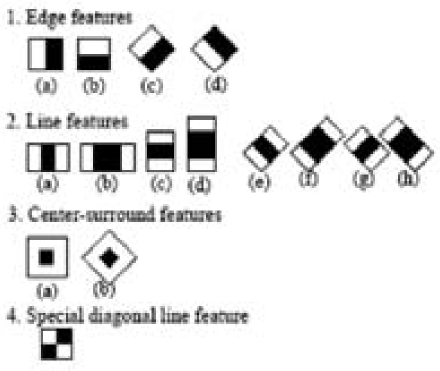

1. Edge features
(a) (b) (c) (d)
2. Line features
(a) (b) (c) (d) (e) (f) (g) (h)
3. Center-surround features
(a) (b)
4. Special diagonal line feature

Figure 2.21: Extended Haar-like feature patterns [LM02].

2.3.2 Dissociated Dipoles

Although the representation of image structure using differential operators that compare adjacent image regions is well-suited to encoding local relationships, such operators have significant drawbacks. One of the most relevant problems appears when we try to compare distant regions of the image. In fact, any comparison of small regions across large distances proves quite difficult, since an operator large enough to span the relevant distance must trade resolution for size. Alternatively, comparing distant regions by propagating information via a chain of small sized operators leads to an increased susceptibility to noise contributed by each of the intermediate elements.

In [BS03], Balas and Sinha introduce the *Dissociated Dipoles* or *Sticks operator*, for encoding non-local image relationships. The aim of that features is to make possible to compare small images across large distances, which adds tolerance to common image transformations. As in the case of Haar-like features, *Dissociated Dipoles* are inspired on a classical wavelet image description technique, the Gabor wavelets, such have been widely applied on iris and fingerprint recognition problems.

Gabor wavelets are closely related to the behavior of the primary visual cortex(see Fig. 2.22). Simple cells in the primary visual cortex have receptive fields which are restricted to small regions of space and highly structured [Mar80]. The first conclusion of the functionality of these cells was described as edge detectors by Hubel and Wiesel [HW59], but later studies performed among others, such as the one by Jones & Palmer [JP87] and De Valois & De Valois [VV88], showed that the response behavior of simple cells of cats corresponds to local measurements of frequencies. Therefore, the interpretation as edge detectors was a first approach of the real properties.

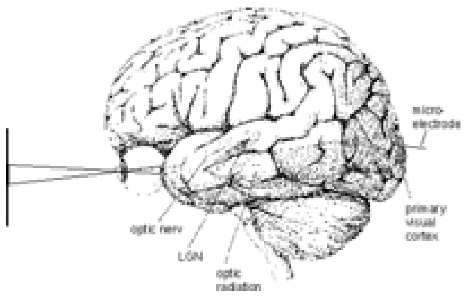


Figure 2.22: The visual pathway in the human brain.

The experiments of Jones & Palmer [JP87] where the receptive field of a certain cells was measured under certain controlled stimulus and the posterior work of Pollen & Ronner [PR81] examining the phase relation of adjacent cells in the visual cortex of cats, concluded that the cells of a pair of adjacent cells have certain symmetries, one of both has even and the other one odd symmetry. This allows to model both receptive fields of such a pair of cells by a complex-valued function, similar to the Gabor wavelets. Therefore, the functionality of the cells in the primary visual cortex can be simulated using Gabor wavelets.

Gabor wavelets are formed from two components, a complex sinusoidal carrier s and a Gaussian envelope w_r. Gabor complex carrier function $s(x, y)$ is defined by equation 2.6 and Fig. 2.23. On the definition of the carrier function, the parameters u_0 and v_0 represent the frequency of the horizontal and vertical sinusoids respectively and P an arbitrary phase shift.

$$s(x, y) = e^{j(2\pi(u_0 x + v_0 y)+P)} \tag{2.6}$$

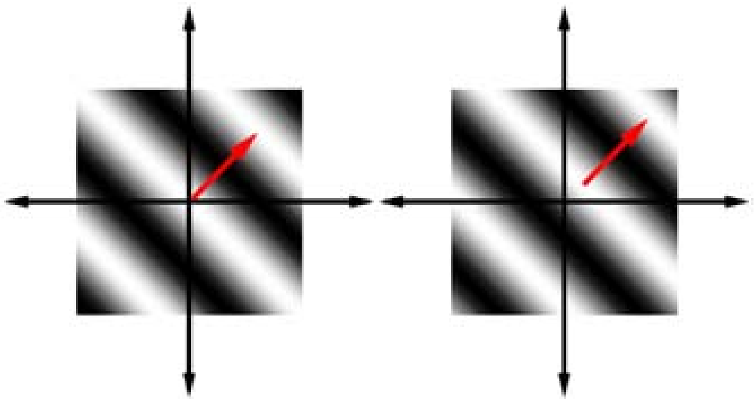


Figure 2.23: The real and imaginary parts of a complex sinusoidal. The images are 128×128 pixels. The parameters are: $u0 = v0 = 1/80$ cycles/pixel, $P = 0$ deg.

Gabor envelop function has a gaussian profile described as:

$$
\begin{aligned}
& w_t(x,y) = Ke^{-\pi\left(a^2(x-x_0)_r^2+b^2(y-y_0)_r^2\right)} \\
& \text{where} \\
& \quad (x-x_0)_r = (x-x_0)cos(\theta) + (y-y_0)sin(\theta) \\
& \quad (y-y_0)_r = -(x-x_0)sin(\theta) + (y-y_0)cos(\theta)
\end{aligned}
\tag{2.7}
$$

where K is a scaling constant, (a,b) the axis scaling constants, θ the rotation constant, and (x_0,y_0) the Gausian envelope peak. The final Gabor wavelet is obtained using the product $g(x,y) = s(x,y) * w_r(x,y)$. An example of 1-D Gabor wavelet is shown in Fig. 2.24.

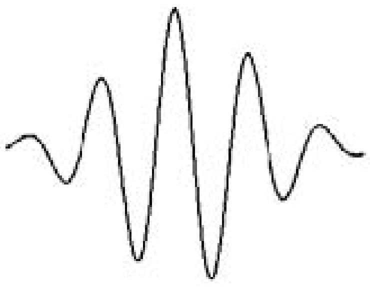


Figure 2.24: Example of a 1-D Gabor wavelet.

The expansion of Gabor wavelets in order to obtain a large enough number of dilations and rotations in order to describe an image may be a very time-consuming task, since this requires computation of biorthogonal wavelets. Therefore, usually, a filter bank consisting of Gabor filters with various scales and rotations is created. A Gabor filter is a linear filter whose impulse response is defined by a harmonic function multiplied by a Gaussian function. Because of the multiplication-convolution property (Convolution theorem), the Fourier transform of a Gabor filter's impulse response is the convolution of the Fourier transform of the harmonic function and the Fourier transform of the Gaussian function. Gabor filters are defined in Eq. 2.8 and Fig. 2.25.

$$
\begin{aligned}
& g(x,y;\lambda,\theta,\psi,\sigma,\gamma) = \exp\left(-\frac{x'^2+\gamma^2y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda}+\psi\right) \\
& \text{with} \\
& \quad x' = x\cos\theta + y\sin\theta \\
& \quad y' = -x\sin\theta + y\cos\theta
\end{aligned}
\tag{2.8}
$$

In this equation, λ represents the wavelength of the cosine factor, θ represents the orientation of the normal to the parallel stripes of a Gabor function, ψ is the phase offset, and γ is the spatial aspect ratio, and specifies the ellipticity of the support of the Gabor function.

While Gabor-like operators provide a simple means of representing image structure, the local image processing they embody limits a recognition system in some significant ways [BS03]:

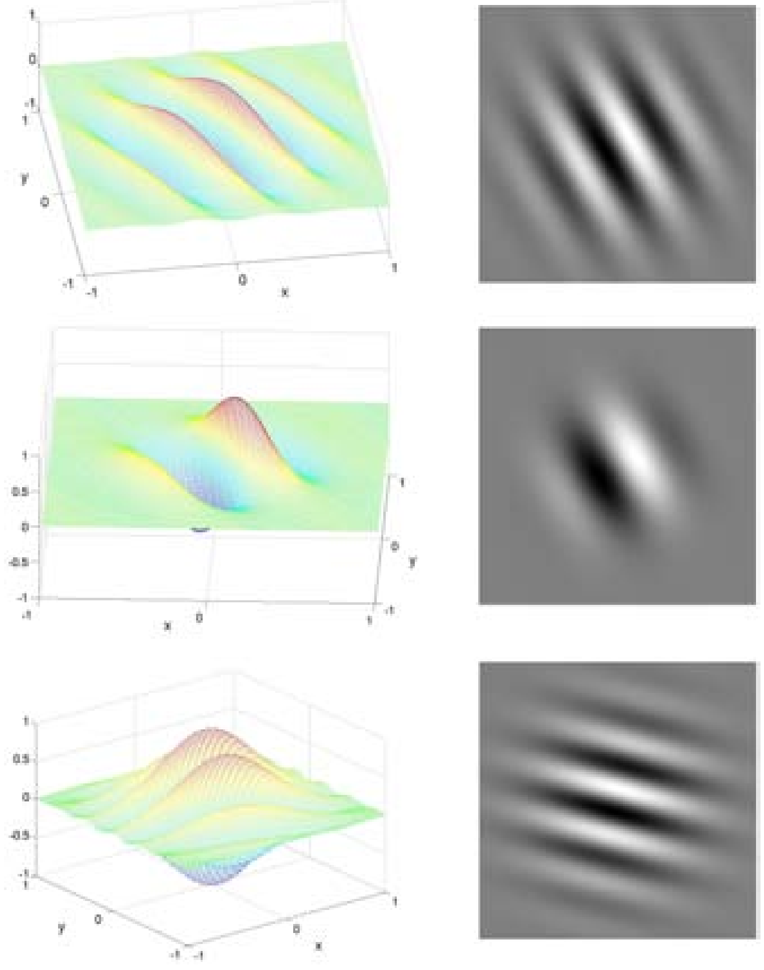


Figure 2.25: Example of Gabor filters with different frequencies and orientations. First column shows their 3D plots and the second one, the intensity plots of their amplitude along the image plane.

Edge-based representations may fail to adapt to small changes in an image brought on by changes in object geometry or position. This particular weakness stems from more general problems with edge-based algorithms, namely that most natural images contain relatively few high-frequency (edge-like) components. Consequently, edge maps implicitly ignore most of the content of an image, and can suffer dramatically from subtle transformations that perturb edge locations while leaving large portions of the image untouched.

Simple analysis also strain the capabilities of a Gabor-based representation scheme due to the conflation of the size of an operator's lobes with the distance spanned by that operator. In fact, any comparison of small regions across large distances

> proves quite difficult, since an operator large enough to span the relevant distance must trade resolution for size. Alternatively, comparing distant regions by propagating information via a chain of small sized operators leads to an increased susceptibility to noise contributed by each of the intermediate elements.

Balas and Shina [BS03] state that the primary source of those shortcomings of the conventional differential operators is the confusion of the inter-lobe distance with lobe-size. Therefore, they de-couple the lobe-size and inter-lobe distance parameters allowing the operator to compare small regions separated by large distances. The result is the *Dissociated Dipole* or *Sticks* operator as a tool for performing non-local image comparisons.

Like a simple edge-finder, a *Dissociated Dipole* is a differential operator consisting of an excitatory and an inhibitory lobe, and may be used at any orientation or scale. However, unlike a conventional edge detector, the correlation of inter-lobe distance and lobe size has been removed, therefore, they allow an arbitrary separation between these two lobes. Formally, the basic form of a *Dissociated Dipole* operator comprises a pair of Gaussian lobes, each with standard deviation σ and a spatial separation of δ. The line joining the centers of the two lobes is at angle θ relative to the horizontal, with θ ranging from 0 to 2π (Fig. 2.26).

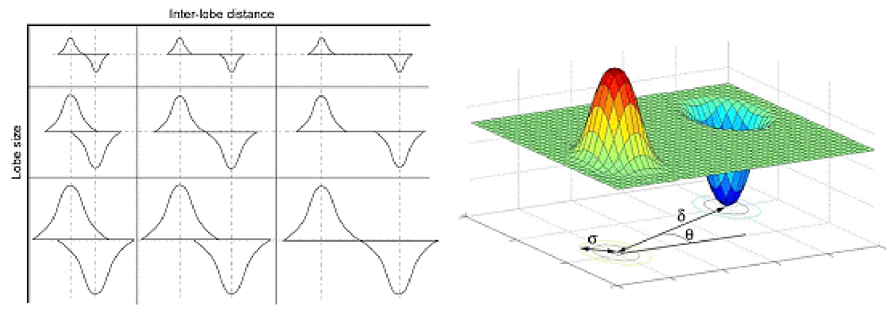


Figure 2.26: At left side, a conventional multi-scale representations that use Gabor-like units confusing the two parameters of inter-lobe distance and lobe size. At right, a schematic representation of a prototypical dissociated dipole [BS03].

As in the case of Haar-like features, *Dissociated Dipole* has a simplified and computationally more feasible representation introduced by Balas & Sinha in [BS06], where the lobes were approximated using rectangles (see Fig. 2.27). The discrimination power of that simplified features was studied from a recognition point of view. Note that some patterns present on Haar-like features shown in Fig. 2.21 can also be simulated with this feature set.

2.4 Pattern Classification

Once phenomena or objects of interest are described using the desired features, the overarching goal and approach is to hypothesize the class of these objects of phenomena, choosing the model that better corresponds to each sensed pattern.

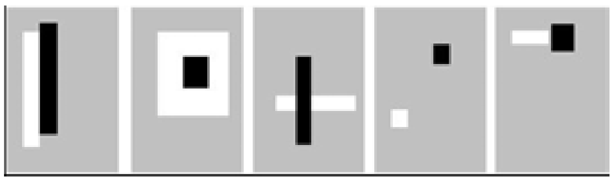


Figure 2.27: Examples of bilobed differential operators of the sort employed in [BS06].

From the beginnings of the pattern classification, an intriguing problem yet to be solved is the relationship between the structure and type of data and the performance of the different classification methodologies. In other words, it is difficult to *a priori* know which is the best approach to be used in a given problem. Although some works such as the one of Van der Walt and Barnard [vdWB06] investigated very specific artificial data sets to determine conditions under which certain classifiers perform better and worse than others, the classical methodology consists of testing the performance of a set of preselected approaches and choose the best one. This previous selection of methodologies usually obeys to problem restrictions such as dimensionality or time/complexity constraints. Determining a suitable classifier for a given problem is however still more an art than a science.

Although it exists a wide range of classification functions, the first approximation to a classification problem should be a linear classifier where the classification decision is based on the value of the linear combination of the features. A linear classifier can be wrote as $y = f'(\mathbf{x}, \mathbf{w}) = g(\mathbf{w} \cdot \mathbf{x} + b)$, where $\mathbf{w}$ is a real vector of weights and g is a function that converts the dot product of the two vectors into the desired output. Often g is a simple function that maps all values above a certain threshold to the first class and all other values to the second class. A more complex g might give the probability that an item belongs to a certain class y.

When working in a binary classification problem, one can visualize the operation of a linear classifier as splitting a high-dimensional input space with a hyperplane, where all points on one side of the hyperplane are classified as *positive*, while the others are classified as *negative*. That type of linear classifiers are usually referred as *decision stumps*, and are often used in situations where the speed of classification is an issue, since it use to be the fastest classifier, especially when $\mathbf{x}$ is sparse or has a large dimensionality. However, decision trees can be faster.

In the literature, one can find two broad classes of methods in order to determine the parameters of a linear classifier, that is, the values of $\mathbf{w}$:

Generative Models Approaches that estimates $\mathbf{w}$ by modeling conditional density functions $P(\mathbf{x}|\text{class})$ (see Fig. 2.28a). Examples of such algorithms include:

Linear Discriminant Analysis (or Fisher's linear discriminant) (LDA), were Gaussian conditional density models are assumed. In contrast to its name, it does not belong to the class of discriminative models in this taxonomy. However, its name makes sense when we compare LDA to the other main linear dimensionality reduction algorithm, such as Principal Components Analysis (PCA).

Naive Bayes classifier which assumes independent binomial conditional density models.

Discriminative Models The discriminative models, which attempt to maximize the quality of the output on a training set (see Fig. 2.28b). Additional terms in the training cost function can easily perform regularization of the final model. Examples of discriminative training of linear classifiers include

Logistic regression A maximum likelihood estimation of **w** is performed assuming that the observed training set was generated by a binomial model that depends on the output of the classifier. In [Mit97], Mitchell shows an interesting relationship between the the logistic regression and Naive Bayes Classifiers: The parametric form of $P(Y|X)$ used by Logistic Regression is precisely the form implied by the assumptions of a Gaussian Naive Bayes classifier. Therefore, we can view Logistic Regression as a closely related alternative to GNB, though the two can produce different results in many cases.

Perceptron An algorithm that attempts to fix all errors encountered in the training set. It is one of the most simple algorithms. Although Minsky & Papert [MP88] caused a significant decline in interest and funding of neural network research, and thus the use of Perceptron, it is studied as one of the first classifiers.

Support vector machine An algorithm that maximizes the margin between the decision hyperplane and the examples in the training set. SVM is considered one of the most relevant methods into the recent start-of-the-art in classification.

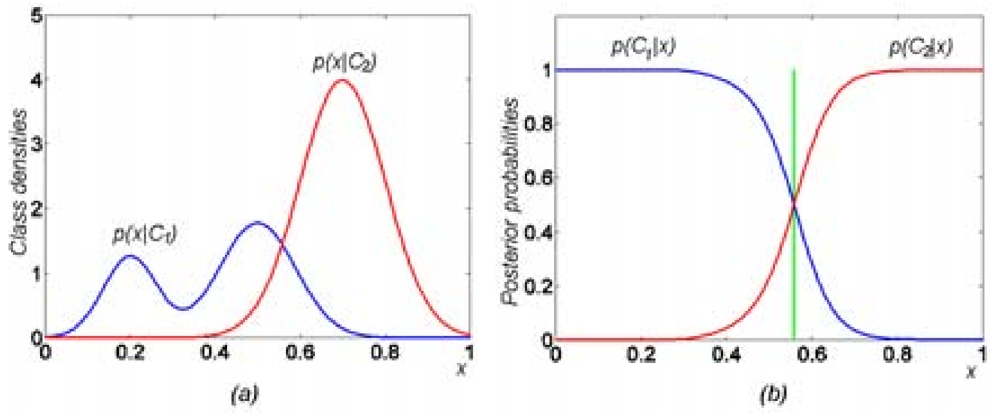


Figure 2.28: Graphical comparison between: *(a)* Generative method and *(b)* Discriminative method [FFFT07].

All of the linear classifier algorithms listed above can be converted into non-linear algorithms operating on a different input space $\varphi(\vec{x})$, using the kernel trick.

Discriminative training often yields higher accuracy than modeling the conditional density functions. However, when training data is limited or when handling missing data generative approaches might be preferable [NJ02].

Recently, a generation of hybrid methods have been developed, where generative models and discriminative learning are intermixed in order to benefit from the two worlds. Examples of those methods can be found in [JH99, HWH05, FLCS05, PD07, BZM08].

Once some of the most classical pattern classification approaches are introduced, next section is related to the general theoretical framework to machine learning, the PAC model of learning. This theory must be understood as the underling idea of most learning methods, and will be used in order to analyze future methods in the thesis.

2.5 PAC Model of Learning

The *Probably Approximately Correct* (PAC) model of learning, is a theoretical framework for studying machine learning. In this section we give only the main definitions for this model, in order to state the basis for studying some learning approaches in further sections.

Definition 1 *Let X be a set called the **instance space**. We think of X as being a set of encodings of instances or objects in the learner's world.*

Definition 2 *A **concept** over X is just a subset $c \subseteq X$ of the instance space. It can be thought of as the set of all instances that positively exemplify some simple or interesting rule. We can equivalently define a concept to be a boolean mapping $c : X \to \{0, 1\}$, with $c(x) = 1$ indicating that x is a positive example of c and $c(x) = 0$ indicating that x is a negative example. For this reason, X is also called the **input space**.*

A **concept class** C over X is a collection of concepts over X. Ideally, we are interested in concept classes that are sufficiently expressive for fairly general knowledge representation.

In this model, a learning algorithm will have access to positive and negative examples of an unknown **target concept** c, chosen from a known concept class C. The learning algorithm will be judged by its ability to identify a hypothesis concept that can accurately classify instances as positive or negative examples of c. In this model the learning algorithms "know" the target class C, in the sense that the designer of the learning algorithm is guaranteed that the target concept will be chosen from C, although it must design the algorithm to work for any $c \in C$.

Definition 3 *Let D be any fixed probability distribution over the instance space X. We will refer to D as the **target distribution**. If h is any concept over X, then the distribution D provides a natural measure of **error** between h and the target concept c. We can define:*

$$error(h) = \mathbf{Pr}_{x \in D}\left[c(x) \neq h(x)\right] \tag{2.9}$$

Definition 4 *Let $EX(c, D)$ be a procedure (sometimes it is called an* oracle*) that runs in unit time, and on each call returns a labelled example $\langle x, c(x)\rangle$, where x is given randomly and independently according to D. A learning algorithm will have access to this procedure when learning the target concept $c \in C$. Ideally, the learning algorithm will satisfy three properties:*

- *The number of calls to $EX(c, D)$ is small, in the sense that it is bounded by a fixed polynomial in some parameters to be specified shortly.*
- *The amount of computation performed is small.*
- *The algorithm outputs a* ***hypothesis concept*** *h such that $error(h)$ is small.*

The number of calls made by a learning algorithm to $EX(c, D)$ is bounded by the running time of the learning algorithm.

Finally, we can define the PAC model as follows:

Definition 5 *Let C be a concept class over X. We say that C is* ***PAC learnable*** *if there exists an algorithm L with the following property: for every concept $c \in C$, for every distribution D on X, and for all $0 < \epsilon < 1/2$ and $0 < \delta < 1/2$, if L is given access to $EX(c, D)$ and inputs ϵ (error parameter) and δ (confidence parameter), then with probability at least $1 - \delta$, L outputs a hypothesis concept $h \in C$ satisfying $error(h) \leq \epsilon$. This probability is taken over the random examples obtained by calls to $EX(c, D)$, and any internal randomization of L.*

The hypothesis $h \in C$ of PAC learning algorithm is thus "approximately correct" with high probability, hence the name Probably Approximately Correct learning. For a more detailed and extended definitions of the PAC model of learning, a good reference is [KV94].

2.6 Ensemble of classifiers

Learning algorithms that output only a single hypothesis suffer from three problems [Die02]:

Statistical Problem: The statistical problem arises when the learning algorithm is searching a space of hypotheses that is too large for the amount of available training data. In such cases, there may be several different hypotheses that all give the same accuracy on the training data, and the learning algorithm must choose one of these to output. There is a risk that the chosen hypothesis will not predict future data points well. When a learning algorithm suffers from this problem, is said to have high "variance".

Computational Problem: The computational problem arises when the learning algorithm cannot guarantee to find the best hypothesis within the hypothesis space. In some types of classifiers, such as neural networks, of decision trees, the task of finding the hypothesis that best fits the training data is computationally intractable, therefore, heuristic methods must be employed. These heuristics can get stuck in local minima, failing to find the best hypothesis. When a learning algorithm suffers from this problem, is said to have high "computational variance".

Representation Problem: The representation problem arises when the hypothesis space does not contain any hypotheses that are good approximations to the true function $f(x)$. When a learning algorithm suffers from this problem, is said to have high "bias".

All those problems can be smoothed by using a weighted vote of hypotheses. A weighted vote of equally accuracy hypotheses reduce the risk in the case of high variance methods. In the same way, considering the combination of several different local minima reduce the risk of choosing the wrong local minimum to output in methods with high computational variance. Finally, the combination of several hypotheses allows to form a more accurate approximation to $f(x)$, improving the methods with high bias. A basic schemata for an ensemble of classifiers that perform a weighted vote is shown in Fig. 2.29. Notice that the final hypothesis h is obtained by combining classifiers hypotheses h_i.

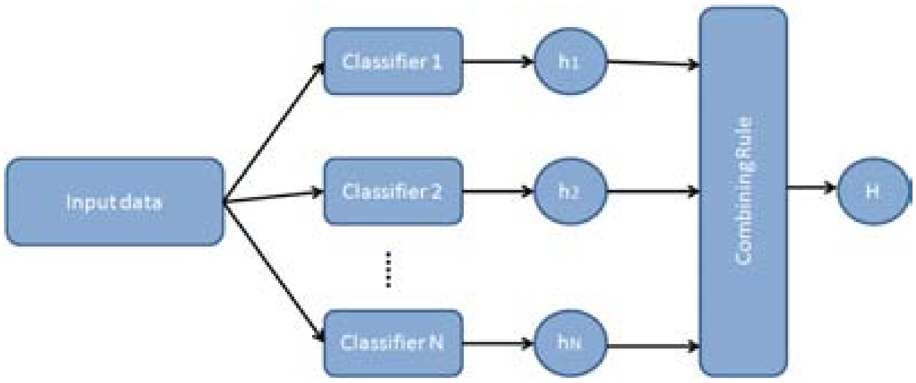


Figure 2.29: Ensemble of classifiers framework.

Ensemble learning algorithms work by running multiple times a given *base learning algorithm*. Depending on how these runs are performed, we can define two different approaches:

Independently Constructed Ensembles: Each hypothesis is constructed independently, in such way that the resulting set of hypotheses individually have a reasonable low error rate for making new predictions and yet the hypotheses disagree with each other in many of their predictions. Therefore, if such an ensemble of those hypotheses can be constructed, it will be more accurate than any of its component classifiers, because the disagreements will cancel out. There are at least four ways to ensure a good set of hypotheses:

- One way to force a learning algorithm to construct multiple hypotheses is to run the algorithm several times and provide it with different training data in each run. The most representative algorithm for this approach is the *Bootstrap Aggregating* or Bagging, introduced by Breiman in [Bre96]. The main idea of Bagging, is to resample the data set multiple times in order to learn different hypotheses, thus, if the learning algorithm is unstable (small changes in training data lead to large changes on resulting hypothesis), the Bagging algorithm will produce a diverse ensemble of hypotheses.

- A second way to produce diverse hypotheses, is to provide a different subset of the input features in each call to the learning algorithm. An example of this approach can be revised in [Che96].
- A third method consists of manipulating the output labels of the training data. This method is known as *error-correcting output coding*, and was proposed by Dietterich and Bakiri in [DB95]. This approach is used when the number of output classes is large, and consists on rewrite the multi-class problem into several binary problems by randomly partitioning the classes into two groups.
- A fourth way to generate accurate and diverse ensembles is to inject randomness into the learning algorithm. An example for this approach in the case of decision trees is shown in [Die00], adding randomness to the process of choosing which feature and threshold to split on. In [Ho98] introduced the *random subspace method* for growing collections of decision trees (*random decision forests*).
- Finally, we can find combination of methods, such as the work of Breiman [Bre01], where Bagging was combined with the random subspace method to grow random decision forests that give excellent performance.

Coordinated Construction of Ensembles: This approach directly addresses the representational problem discussed above. It consists to construct the hypotheses in a coupled fashion so that the weighted vote of the hypotheses gives a good fit to the data. The most used approach in this case is the so-called *Boosting*. *Boosting* can be seen as a type of *Bagging*, where instead of sample the data set, in this case a distribution over the examples is used in order to construct complementary hypotheses. This method and some classical algorithms are analyzed on future sections.

After a general overview of the ensembles of classifiers and how to learn them, now we will concentrate only on the *Boosting* approach and the *Adaptive Boosting* (AdaBoost) [FS96], one of the most used *Boosting* algorithms. We first present a theoretical framework for *Boosting* to analyze deeply the *AdaBoost* algorithms and their variants.

2.6.1 Boosting

Boosting is a general method for improving the accuracy of any given learning algorithm. In the following, some new definitions based on the introduction to the PAC Model of Learning in the section 2.5 are introduced:

Definition 6 *Given ϵ, $\delta > 0$ and access to random examples, an algorithm is a* strong *PAC-learning algorithm if outputs with probability $1 - \delta$ a hypothesis with error at most ϵ. Further, the running time must be polynomial in $1/\epsilon$, $1/\delta$ and other relevant parameters (namely, the "size" of the examples received, and the "size" or "complexity" of the target concept).*

Definition 7 *Given ϵ, $\delta > 0$ and access to random examples, an algorithm is a* weak *PAC-learning algorithm if outputs with probability $1 - \delta$ a hypothesis with error at most $\epsilon \geq 1/2 - \gamma$, where $\gamma > 0$ is either a constant, or decreases as $1/p$ where p is a polynomial in the relevant parameters [KV94, FS97]. We will use* ***WeakLearn*** *to denote a generic weak learning algorithm.*

In [Sch90], Schapire showed that any weak learning algorithm can be efficiently transformed or "boosted" into a strong learning algorithm. Later, in [Fre95] Freund presented the "boost-by-majority" algorithm that is considerably more efficient than Schapire's. Both algorithms work calling a given weak learning algorithm **WeakLearn** multiple times, each time with a different distribution over X, and finally combining all of the generated hypotheses into a single hypothesis. The intuitive idea is to alter the distribution over the domain X in a way that increases the probability of the "harder" parts of the space, thus forcing the weak learner to generate new hypotheses that make less mistakes on these parts.

Boost-by-majority algorithm requires that the bias γ of the weak learning algorithm **WeakLearn** be known ahead of time, and it is an important practical deficiency. Not only is this worst-case bias usually unknown in practice, but the bias that can be archived by **WeakLearn** will typically vary considerably from one distribution to the next. Unfortunately, the boost-by-majority algorithm cannot take advantage of hypotheses computed by **WeakLearn** with error significantly smaller that the presumed worst-case bias of $1/2 - \gamma$.

2.6.2 Adaboost

Adaboost algorithm, introduced by Freund and Schapire [FS97] is very nearly as efficient as boost-by-majority. However, unlike boost-by-majority, the accuracy of the final hypothesis produced by Adaboost depends on the accuracy of all the hypotheses returned by **WeakLearn**, and so is able to more fully exploit the power of the weak learning algorithm [Fre95].

Although boosting has its root in the PAC model, Adaboost is defined in a more general learning framework, in which the learner receives examples (x_i, y_i) chosen randomly according to some fixed but unknown distribution P on $X \times Y$, where Y is a set of possible labels. As usual, the goal is to learn to predict the label y given an instance x. Adaboost calls a given weak learning algorithm repeatedly in a series of rounds $t = 1, ..., T$. One of the main ideas of the algorithm is to maintain a distribution or set of weights over the training set. The weight of this distribution on training example i on round t is denoted $D_t(i)$. Initially, all weights are set equally, but on each round, the weights of incorrectly classified examples are increased so that the weak learner is forced to focus on the hard examples in the training set.

The goodness of a weak hypothesis is measured by its *weighted error*

$$\epsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(x_i)\neq y_i} D_t(i) \tag{2.10}$$

Alternatively, when the weak learner cannot be trained using the weights D_t on the training examples, a subset of the training examples can be sampled according to D_t, and used to train the weak learner.

At each iteration, Adaboost calculates the updating rule β_t. The parameter β_t is chosen as a function of ϵ_t and is used for updating the D_t. The update rule reduces the probability assigned to those examples on which the hypothesis makes a good prediction and increases the probability of the examples on which the prediction is poor. Furthermore, if h_t is Boolean (with range $\{0,1\}$), then it can be shown that this update rule exactly removes the advantage of the last hypothesis. That is, the error of h_t on distribution D_{t+1} is exactly $\frac{1}{2}$. The original Adaboost algorithm proposed by Freund and Shcapire [FS97] is shown in Algorithm 1.

Algorithm 1 The adaptive boosting algorithm [FS96]

```
Input: A training set $X$ of $N$ pairs $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ is the $i^{th}$ image and $y_i \in \{0,1\}$ is the category of the object present in $\mathbf{x}_i$, a weak learning algorithm (WeakLearner), a distribution $D$ over the $N$ examples, and the maximum number of iterations $T$.
  Initialize the weight vector: $w_i^1 = D(i) \forall i = 1, ..., N$.
  for $t = 1, .., T$ do
    Set
        $p^t = \frac{w^t}{\sum_{i=1}^N w_i^t}$
    Call WeakLearn, providing it with the distribution $\mathbf{p^t}$: get back a hypothesis $h_t : X \to [0,1]$.
    Calculate the error of $h_t : \epsilon_t = \sum_{i=1}^N p_i^t |h_t(x_i) - h_i|$
    Set $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
    Set the new weights vector to be
        $w_i^{t+1} = w_i^t \beta_t^{1-|h_t(x_i)-y_i|}$
  end for
Output: The final hypothesis:
    $H(x) = \begin{cases} 1 & if \quad \sum_{t=1}^T \left(\log \frac{1}{\beta_t}\right) h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \log \frac{1}{\beta_t} \\ 0 & otherwise \end{cases}$
```

The Adaboost algorithm can be interpreted as an stagewise estimation procedure for fitting an additive logistic regression model. It optimizes an exponential criterion which to second order is equivalent to the binomial log-likelihood criterion.

In [FHT00], Friedman proposes a more standard likelihood-based boosting procedure. Because this algorithm will be widely used in this thesis, a theoretical overview based on the Friedman's work is introduced.

Viewing the boosting procedures as a stagewise algorithms for fitting additive models helps to understand their performance. AdaBoost fits an additive model $F(x) = \sum_{m=1}^{M} c_m f_m(x)$.

We initially focus on the regression problem, where the response y is quantitative, x and y have some joint distribution, and we are interested in modelling the mean $E(y|x) = F(x)$. The additive model has the form

$$F(x) = \sum_{j=1}^{p} f_j(x_j) \tag{2.11}$$

There is a separate function $f_j(x_j)$ for each of the p input variables x_j. More generally, each component f_j is a function of a small, prespecified subset of the input variables. The *backfitting algorithm* [FS81, BHT89] is a convenient modular "Gauss-Seidel" algorithm for fitting additive models. A backfitting update is

$$f_j(x_j) \leftarrow E\left[y - \sum_{k \neq j} f_k(x_k)|x_j\right] for j = 1, 2, ..., p, 1, ... \tag{2.12}$$

Any method or algorithm for estimating a function of x_j can be used to obtain an estimate of the conditional expectation in Eq. 2.12. This can include nonparametric smoothing algorithms, such as local regression or smoothing splines. In the right-hand side, all the latest versions of the functions f_k are used in forming the partial residuals. The backfitting cycles are repeated until convergence. Under fairly general conditions, backfitting can be shown to converge to the minimizer of $E(y - F(x))^2$ [BHT89].

Now we will consider an additive model whose elements $\{f_m(x)\}_1^m$ are functions of potentially all of the input features x. In this context the $f_m(x)$ are taken to be simple functions characterized by a set of parameters γ and a multiplier β_m,

$$f_m(x) = \beta_m b(x; \gamma_m) \tag{2.13}$$

The additive model then becomes

$$F_M(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m) \tag{2.14}$$

If least-squares is used as a fitting criterion, one can solve for an optimal set of parameters through a generalized backfitting algorithm with updates,

$$\{\beta_m, \gamma_m\} \leftarrow \arg\min_{\beta,\gamma} E\left[y - \sum_{k \neq m} \beta_k b(x; \gamma_k) - \beta b(x; \gamma)\right]^2 \tag{2.15}$$

for $m = 1, 2, ...M$ in cycles until convergence. Alternatively, one can use a "greedy" forward stepwise approach,

$$\{\beta_m, \gamma_m\} \leftarrow \arg\min_{\beta,\gamma} E\left[y - F_{m-1}(x) - \beta b(x; \gamma)\right]^2 \tag{2.16}$$

for $m = 1, 2, ..., M$ where $\{\beta_k, \gamma_k\}_i^{m-1}$ are fixed at their corresponding solution values at earlier iterations.

In boosting jargon, $f(x) = \beta b(x; \gamma)$ would be called a **weak learner** and $F_M(x)$ (Eq. 2.14) the **committee**. Note in this point, that the backfitting procedure, independently of the version we use, the general or the greedy, only require an algorithm for fitting a *single* weak learner (Eq. 2.13) to data. This base algorithm is simply applied repeatedly to modified versions of the original data

$$y_m \leftarrow y - \sum_{k \neq m} f_k(x) \tag{2.17}$$

In the forward stepwise procedure (Eq. 2.16), the modified output y_m at the mth iteration depends only on its value y_{m-1} and the solution $f_{m-1}(x)$ at the previous iteration,

$$y_m = y_{m-1} - f_{m-1}(x) \tag{2.18}$$

At each step m, the previous output value y_{m-1} is modified so that the previous model f_{m-1} has no explanatory power on the new outputs y_m. One can therefore view this as a procedure for boosting a weak learner $f(x) = \beta b(x; \gamma)$ to form a powerful committee $F_M(x)$.

Now, consider minimizing the criterion:

$$J(F) = E\left(e^{-yF(x)}\right) \tag{2.19}$$

for estimation of F(x). Here E represents expectation: depending on the context, this may be a population expectation (with respect to a probability distribution) or else a sample average. E_w indicates a weighted expectation. Lemma 1 shows that the function $F(x)$ that minimizes $J(F)$ is the symmetric logistic transform of $P(y = 1|x)$. (The proof can be found in [FHT00]).

Lemma 1 *$E(e^{-yF(x)})$ is minimized at*

$$F(x) = \frac{1}{2} \log \frac{P(y = 1|x)}{P(y = -1|x)} \tag{2.20}$$

Hence

$$P(y = 1|x) = \frac{e^{F(x)}}{e^{-F(x)} + e^{F(x)}} \tag{2.21}$$

$$P(y = -1|x) = \frac{e^{-F(x)}}{e^{-F(x)} + e^{F(x)}} \tag{2.22}$$

Corollary 2 *If E is replaced by averages over regions of x where $F(x)$ is constant (as in the terminal node of a decision tree), the same result applies to the sample proportions*

Result 3 *The* Discrete Adaboost *algorithm builds an additive logistic regression model via Newton-like updates for minimizing $E(e^{-yF(x)})$*

Adaboost variants

Since Freund and Shcapire presents their AdaBoost algorithm, many other versions of this algorithm have been developed. All these versions differ on the manner how they modifies the weights and construct the final hypothesis. In the following sections we show the algorithms for some of that versions. As in the original AdaBoost algorithm, all their versions can be interpreted from a statistical point of view [FHT00]. In the following, the key points and algorithms of some different versions are introduced. Notice that the versions are slightly different in the way they update the weights and in the output formulation.

Real Adaboost The *Real AdaBoost* algorithm builds an additive logistic regression by stagewise and approximate optimization of $J(F) = E\left[e^{-yF(x)}\right]$. The Real Adaboost can be viewed as an Adaboost with Confidence Weighted Predictions, in the sense that it uses class probability estimates $p_m(x)$ to construct real-valued contributions $f_m(x)$. The algorithm is showed in Algorithm. 2.

Algorithm 2 The Real AdaBoost algorithm

```
Input: A training set $X$ of $N$ pairs $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ is the $i^{th}$ image and $y_i \in \{0,1\}$ is
  the category of the object present in $\mathbf{x}_i$, a weak learning algorithm (WeakLearner)
  and the maximum number of iterations $T$.
  Initialize the weight vector $w_i = \frac{1}{N}$, $i = 1, 2, ..., N$
  for $t = 1, .., T$ do
    Use the WeakLearn in order to fit the classifier to obtain a class probability
    estimate $p_t(x) = \hat{P}_w(y = 1|x) \in [0, 1]$, using weights $w_i$ on the training data
    Set $f_t(x) \leftarrow \frac{1}{2} \log \frac{p_t(x)}{1-p_t(x)} \in \mathbf{R}$
    Set $w_i \leftarrow w_i e^{-y_i f_t(x_i)}$, $i = 1, 2, ..., N$, and renormalize so that $\sum_i w_i = 1$.
  end for
Output: The final hypothesis:
```

$$F_T = sign\left(\sum_{t=1}^{T} f_t(x)\right)$$

LogitAdaboost The *LogitAdaboost* algorithm uses Newton steps for fitting an additive symmetric logistic model by maximum likelihood. The algorithm is showed in Algorithm. 3.

Gentle Adaboost The *Gentle Adaboost* algorithm uses Newton steps for minimizing $E\left[e^{-yF(x)}\right]$. This is a modified version of the *Real Adaboost*, using Newton stepping rather than exact optimization at each step. The algorithm is shown in Algorithm 4.

In addition to the different Adaboost variants based on the way they use the error values to update the weights, in the literature we can find different modifications in order to improve other aspects of the obtained classifiers. For instance, Mita [MKSH08]

Algorithm 3 The Logit AdaBoost algorithm

```
Input: A training set $X$ of $N$ pairs $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ is the $i^{th}$ image and $y_i \in \{0,1\}$ is
    the category of the object present in $\mathbf{x}_i$, a weak learning algorithm (WeakLearner)
    and the maximum number of iterations $T$.
  Start with weights $w_i = \frac{1}{N}$, $i = 1,2,...,N$, $F(x) = 0$ and probability estimates
    $p(x_i) = \frac{1}{2}$
  for $t = 1,..,T$ do
    Compute the working response
        $$z_i = \frac{y_i^* - p(x_i)}{p(x_i)(1 - p(x_i))}$$
    where $y_i^*$ represents outcome and it's a 0/1 response.
    Compute the weights
        $$w_i = p(x_i)(1 - p(x_i)$$
    Find the function $f_t(x)$ by a weighted least-squares regression of $z_i$ to $x_i$ using
      weights $w_i$.
    Update $F(x) \leftarrow F(x) + \frac{1}{2} f_t(x)$ and $p(x) \leftarrow \frac{e^{F(x)}}{e^{F(x)}+e^{-F(x)}}$
  end for
Output: The final hypothesis:
        $$F_T = sign[F(x)] = sign\left[\sum_{t=1}^{T} f_t(x)\right]$$
```

improves the generalization performance using weak classifiers that include multiple features simultaneously. Feature co-occurrence makes it possible to classify difficult samples that are misclassified by weak classifiers using a single feature.

2.7 An approach to object detection

After introducing the object detection problem and all their components from a theoretical point of view, in this section a more practical view of the Viola & Jones face detector [VJ01] is introduced as an example of a successful object detection approach. This approach use the Adaboost algorithm in order to learn ensembles of Haar-like based decision stumps. In addition to their good performance, this approach is interesting from a practical point of view due to their real time ability, which is archived using the combination of a cascaded architecture with an optimal calculation of the Haar-like features.

Algorithm 4 The Gentle AdaBoost algorithm

```
Input: A training set $X$ of $N$ pairs $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ is the $i^{th}$ image and $y_i \in \{0,1\}$ is
  the category of the object present in $\mathbf{x}_i$, a weak learning algorithm (WeakLearner)
  and the maximum number of iterations $T$.
  Initialize the weight vector $w_i = \frac{1}{N}$, $i = 1, 2, ..., N$
  Initialize $F(x) = 0$
  for $t = 1, .., T$ do
    Fit the regression function $f_t$ by weighted least-squares of $y_i$ to $x_i$ with weights
    $w_i$.
    Update $F(x) \leftarrow F(x) + f_t(x)$
    Update $w_i \leftarrow w_i e^{-y_i f_t(x_i)}$ and renormalize.
  end for
Output: The final hypothesis:
```

$$F_T = sign[F(x)] = sign\left[\sum_{t=1}^{T} f_t(x)\right]$$

2.7.1 Detection Process

The detection scheme corresponds to a *Rare event detection* problem (see Section 2.1.2). During the system learning stage, it is necessary to define a *learning window size*, which is the reference size of the objects. During the detection process, the input image is decomposed in a huge set of overlapping regions, which must be proportional to the *learning window size*, allowing a multi scale detection. All these regions are then classified as object or non-object (see Fig. 2.7.1).

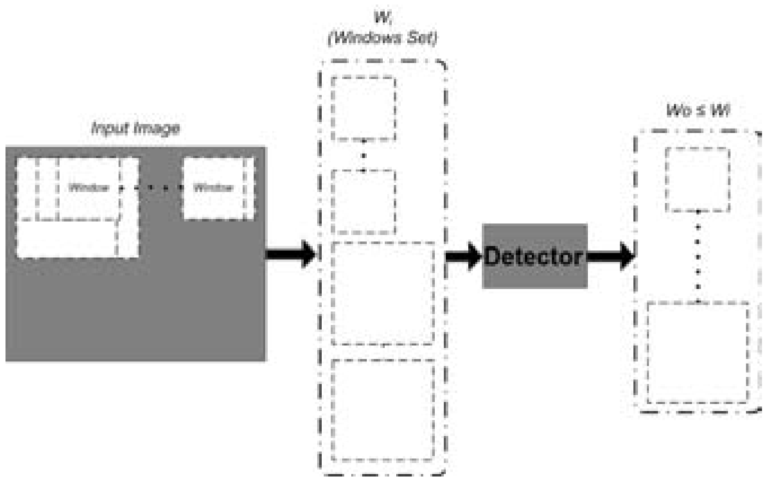


Figure 2.30: Detection process using a scanning of the input image.

As we discuss before, when a *rare event detection* problem is faced, there are hard restrictions on which classification methods are used, and fast methods are required. Next sections presents the different ingredients of the classification system. Firstly, as it is used the AdaBoost algorithm, the **WeakLearn** function is defined. Later, an efficient calculation and normalization of features is addressed and finally, a strategy to speed-up the detection process is presented.

2.7.2 Weak learner

As we viewed on Section 2.6.2, Adaboost needs of an auxiliary process, which is named **WeakLearn** to learn the classification rule. This section deals with the weak learner algorithm.

If we define a *weak classifier* as function $h : X \mapsto \{-1, 1\}$ that

$$h_{c,p,thr}(x) = \begin{cases} 1 & if \quad c(x) \times p \geq thr \times p \\ -1 & otherwise \end{cases} \tag{2.23}$$

where $c(x)$ is the result to calculate the feature c of the features set on an input image x, $p \in \{-1, 1\}$ is the polarity value which indicates if the positive values are over or under the threshold value and $thr \in \mathbf{R}$ is the threshold value.

Given a samples set $X = \langle(x_1, y_1), \ldots, (x_N, y_N)\rangle$, and their associated weights $\{w_1, \ldots, w_N\}$, the WeakLearn's objective is to find the best feature and their parameters. This is, find the *weak classifier* that minimizes the *weighted classification error*

$$\epsilon = \sum_{i:h(x_i) \neq y_i} w_i \tag{2.24}$$

This process is done by the greedy algorithm shown in Algorithm 5. Notice in this algorithm, that the learning time for a simple *weak classifier* involves a large amount of calculation. The time that the *WeakLearn* needs to find the best hypothesis depends on the number of features in the feature set and in the number of samples in the training set. This point will be revisited in the future.

2.7.3 Integral Images

Although Viola & Jones work used a simple Haar-like feature set, the classical implementation for their approach is based on the extended Haar-like feature set introduced by Lienhart and Maydt in [LM02] (see Fig. 2.21). Haar features present an interesting property in the context of object recognition: they can be computed very fast and in constant time for any size by means of two auxiliary images. In the case of horizontal and vertical oriented patterns the auxiliary image is the *Summed Area Table* (SAT), and for 45^o rotated patterns the *Rotated Summed Area Table* ($RSAT$). Those images are also known as *Integral Images*.

The $SAT(x, y)$ is defined as the sum of the pixels of the upright rectangle ranging from the top left corner to the bottom right corner at (x, y) (see Fig. 2.7.3a). If I is the input image, we can define

$$SAT(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y')$$

Algorithm 5 Greedy WeakLearn algorithm.

```
Input: A training set $X$ of $N$ pairs $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ is the $i^{th}$ image and $y_i \in \{0,1\}$
  is the category of the object present in $\mathbf{x}_i$, a feature set $\mathfrak{C}$, and a weights distribution
  $D = \{w_1, ...w_N\}$ over the $N$ examples.
  Initialize $E(i) = 1.0$, $P(i) = 1.0$, and $T(i) = 0.0$ for $i = 1, ..., N$,
  $i \leftarrow 1$
  for all $c \in \mathfrak{C}$ do
    Get $Z = \{c(x_i) | i = 1, ..., N\}$
    Create $Z'$ with one instance of all the different values of $Z$, without repetitions.
    Sort in ascending way the set $Z'$.
    Create a set of all the possible threshold:
      $$\mathfrak{T} = \left\{ -\infty, \frac{Z'(2) - Z'(1)}{2}, ..., \frac{Z'(i+1) - Z'(i)}{2}, ..., \infty \right\}$$
    for all $t \in \mathfrak{T}$ do
      Generate a hypothesis for each sample as:
        $$h(x) = \begin{cases} 1 & if \quad c(x) \geq t \\ -1 & otherwise \end{cases}$$
      Calculate the error
        $$\epsilon = \sum_{i:h(x_i) \neq y_i} w_i$$
      if $\epsilon < 0.5$ then
        $\epsilon^* \leftarrow \epsilon$ and $p^* \leftarrow 1.0$
      else
        $\epsilon^* \leftarrow 1.0 - \epsilon$ and $p^* \leftarrow -1.0$
      end if
      if $\epsilon^* < E(i)$ then
        $E(i) \leftarrow \epsilon^*$, $P(i) \leftarrow p^*$, and $T(i) \leftarrow t$
      end if
    end for
    $i \leftarrow i + 1$
  end for
Output: The best hypothesis
    $$h_{c,p,t}(x) = \begin{cases} 1 & if \quad c(x) \times p \geq thr \times p \\ -1 & otherwise \end{cases}$$
  where $c \leftarrow \mathfrak{C}(i)$, $p \leftarrow P(i)$, and $t \leftarrow T(i)$ such as $E(i) \leq E(j) | \forall_{i,j}, i \neq j$
```

It can be calculated with one pass over all pixels from left to right and top to bottom by means of

$$SAT(x,y) = SAT(x,y-1) + SAT(x-1,y) + I(x,y) - SAT(x-1,y-1)$$

with

$$SAT(-1,y) = SAT(x,-1) = SAT(-1,-1) = 0$$

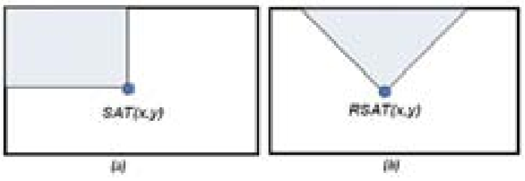


Figure 2.31: Definition of (a) *Summed Area Table* (SAT) (b) *Rotated Summed Area Table* ($RSAT$).

Analogously to SAT, $RSAT(x,y)$ is defined as the sum of the pixels of of a 45^o rotated rectangle with the bottom most corner at (x,y) and extending upwards till the boundaries of the image (see Fig. 2.7.3b).

$$RSAT(x,y) = \sum_{y' \leq y, y' \leq y - |x - x'|} I(x',y')$$

It can be calculated also in one pass from left to right and top to bottom over all pixels by

$$\begin{aligned} RSAT(x,y) = &\ RSAT(x-1,y-1) + RSAT(x+1,y-1) \\ - &\ RSAT(x,y-2) + I(x,y) + I(x,y-1) \end{aligned}$$

with

$$\begin{gathered} RSAT(-1,y) = RSAT(x,-1) = RSAT(x,-2) = 0 \\ RSAT(-1,-1) = RSAT(-1,-2) = 0 \end{gathered}$$

Once both auxiliary images, SAT and $RSAT$, are calculated, any rectangle can be calculated with only 4 accesses to one of these images. If we define the rectangle as $r = (x,y,w,h,\alpha)$, the sum of all the values can be calculated as:

$$\begin{aligned} RecSum(r) = &\ SAT(x-1,y-1) + SAT(x+w-1,y+h-1) \\ - &\ SAT(x-1,y+h-1) - SAT(x+w-1,y-1) \end{aligned}$$

when $\alpha = 0^o$ and

$$\begin{aligned} RecSum(r) = &\ RSAT(x-h+w,y+w+h-1) + RSAT(x,y-1) \\ - &\ RSAT(x-h,y+h-1) - RSAT(x+w,y+w-1) \end{aligned}$$

when $\alpha = 45^o$

Finally, to evaluate a rectangle feature, we just have to decompose it in simple rectangles and evaluate each rectangle independently and finally combine all the values in order to get the feature value.

2.7.4 Image Normalization

Changes on the lighting conditions can alter the feature values, therefore, the hypothesis obtained by the threshold based decision stumps can be altered. In order to get illumination invariance, a normalization method called *fast lighting correction* is used. The special properties of the rectangle features also enable fast contrast stretching of the form

$$\bar{I}(x,y) = \frac{I(x,y) - \mu}{c\sigma}, c \in \mathbf{R}^+ \tag{2.25}$$

μ can easily be determined by means of $SAT(x,y)$. Computing σ, however, involves the sum of squared pixels. It can easily be derived by calculating a second set of SAT and $RSAT$ auxiliary images for $I^2(x,y)$. Then, calculating σ for any window requires only 4 additional table lookups.

Notice that from a practical point of view, the four auxiliary images must be only calculated once for the input image, and are used in the classification of all the subregions. Therefore, the timing cost for this calculation is insignificant in contrast to the classification time.

2.7.5 Attentional Cascade

Although the optimal image normalization and feature calculation approaches, to get good results in a real-world object detection problem, the *strong classifier* learnt by AdaBoost must be a combination of a large number of *weak classifiers.* Since we need to apply this classifier to a huge number of regions, the final detection time for an image is prohibitive. In order to address that limitation, Viola & Jones introduced a cascade architecture of multiple *strong classifiers.* The underlying idea is use only the necessary computation cost in order to reject a non object image, while more complex analysis is performed in more difficult ones (see Fig. 2.32). Those regions that arrives to the last stage of the cascade, and are classified as objects, are selected as object regions, the rest of the regions are rejected.

Each stage analyze only the objects accepted by the previous stages, and thus, the non-objects are analyzed only until they are rejected by a detector. The number of applied classifiers is reduced exponentially due to the cascade architecture. Notice that with a false alarm of 0.5, the half part of the input regions to an stage are rejected, while the other half part pass to the next stage. In addition, the stages are *strong classifiers* with weak performance restrictions, and therefore, the number of *weak classifiers* that conform them is smaller.

Learning an *attentional cascade* is not significantly more complex than learning a unique *strong classifier*, in the sense that it is used the same algorithm, but changing the training set. Adaboost is used to learn each stage of the cascade, and the rejected samples are changed by other non-objects that the previous trained stages classify by correct objects. The training algorithm for the cascade is showed in Algorithm 6. The final false alarm of the detectors cascade will be $F_{target} = f^n$, where f is the false alarm ratio fixed for each stage of the cascade and n is the number of stages. Analogously, the final hit ratio can be estimated as d^n.

There are some aspects from algorithm 6 which are important to highlight. The first one is that the number of negative samples in N needs to be enormous. At

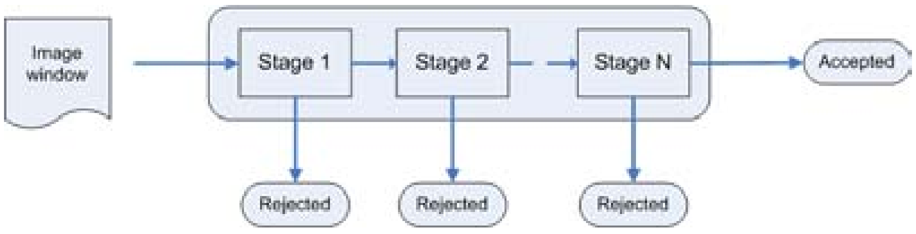


Figure 2.32: The attentional cascade

Algorithm 6 Attentional cascade training algorithm

```
Input: User selected values for $f$, the maximum acceptable false positive rate per
  layer, the minimum acceptable detection rate per layer $d$, and the target overall false
  positive rate $F_{target}$. In addition, the algorithm needs a set of positive examples $P$
  and a set of negative examples $N$.
  Initialize $F_0 = 1.0$ and $D_0 = 1.0$
  Set $i = 1$
  while $F_i > F_{target}$ do
    $i \leftarrow i + 1$
    $n_i = 0; F_i = F_{i-1}$
    while $F_i > f \times F_{i-1}$ do
      $n_i \leftarrow n_i + 1$
      Use $P$ and $N$ to train a classifier with $n_i$ features using Adaboost
      Evaluate current cascaded classifier on validation set to determine $F_i$ and $D_i$
      Decrease threshold for the $i$th classifier until the current cascaded classifier has
      a detection rate of at least $d \times D_{i-1}$ (this also affects $F_i$)
    end while
    $N \leftarrow 0$
    if $F_i > F_{target}$ then
      evaluate the current cascaded detector on the set of non-object images and put
      any false detections into the set $N$.
    end if
  end while
Output: A cascade of strong classifiers.
```

each iteration we need to replace the rejected samples with new samples that are miss-classified by previous stages, and our classifier increases the rejection rate with the number of stages, therefore each time is more difficult to find new samples. The number of negative examples can be estimated as:

$$N^{+} = K + \sum_{i=1}^{S-1} \frac{K \times (1 - f)}{f^{i}} \tag{2.26}$$

where S is the number of stages, K the number of negative samples we use to train a stage and f the maximum false rate per stage. That is, if we decide to learn a cascade of 10 stages with $f = 0.5$ and $K = 1.000$, which are common values, we need a negative samples set of about 1.024.000 samples. To avoid to create training sets of such dimension, usually the negative samples are collected on-line from image databases or either analyzing TV programmes or other large image sources. Summarizing, to learn a cascade of detectors, usually we need to perform the detection process on multiple images.

Chapter 3

Evolutionary computation

Evolutionary computation refers to a wide family of methods that are inspired on the Darwin's evolution theory. Natural evolution is a population-based optimization process. The simulation of this process using computers results in stochastic optimization techniques which often outperform classical methods of optimization when applied to difficult real-world problems. In this chapter, a brief introduction to evolutionary computation and some of their most representative implementations is introduced, based on the concept of Darwin machines. Finally, implementation of Darwin Machines using genes-based approach and different probability-based approaches are analyzed in depth to define the evolutionary framework we need on the next chapter.

3.1 Introduction

Darwin hypothesized that living beings adapted and differentiated to varying conditions or niches in their environment through a process of evolution. Although in Darwin times genetics was an unknown field, most of the works which are based on evolution assumes the actual knowledge about genetics and define algorithms based on a chromosome-based encoding, and the processes observed in natural evolution of species. However, the Darwinian processes can be defined in a more general manner, with no assumptions about the implementation of these processes. This more general definition has been widely developed in memetics [Bla00, Cal97], the field that attempts to cope with evolutionary models of information transmission. These models have been used in order to model some brain functionality and social conducts. In [Cal97], Calvin defines the essential processes in any evolutionary model, in order to ensure a quality improvement along generations:

1. There must be a pattern involved.

2. The pattern must be copied somehow.

3. Variant patterns must sometimes be produced by chance.

4. The pattern and its variant must compete with one another for occupation of a limited work space.

5. The competition is biased by a multifaceted environment. That's Darwin's natural selection.

6. New variants always preferentially occur around the more successful of the current patterns. This is what Darwin later called an inheritance principle.

In addition to these essential processes, Calvin [Cal97] introduce five other processes which can notably influence the rate of evolutionary change:

1. Stability may occur. Variants happen, but they're either nonviable or backslide easily.

2. Systematic recombination (crossing over, sex) generates many more variants than do copying errors and the far-rarer point mutations.

3. Fluctuating environments, shaping up more complex patterns capable of doing well in several environments.

4. Parcellation typically speeds evolution. It raises the surface-to-volume ratio (or perimeter-to-area ratio) and exposes a higher percentage of the population to the marginal conditions on the margins.

5. Local extinctions speed evolution because they create empty niches.

Once Darwinian processes have been introduced, Calvin [Cal87] defines a *Darwin machine* by analogy to a Turing machine, as a machine that, like a Turing machine, involves an iteration process that yields a high-quality result, but, whereas a Turing machine uses logic, the Darwin machine uses rounds of variation, selection, and inheritance. In its original connotation, a *Darwin machine* is any process that bootstraps quality by utilizing all of the six essential features of a Darwinian process: A pattern is copied with variations, where populations of one variant pattern compete with another population, their relative success biased by a multifaceted environment so that winners predominate in producing the further variants of the next generation.

The theoretical definition of a *Darwin machine* states which are the necessary conditions for evolution, and there are different possibilities to implement these processes. The most widely applied implementation try to copy the natural implementation of those processes, using genetic theories, and is what we refer as *Genetic Darwin Machine*. Another possibility is to use probabilistic models in order to implement these processes. Although in literature these methods can be found under a wide set of different names, we group all these approaches under the name of *Probabilistic Darwin Machines*. We first define the basic concepts for both implementations and next sections deeply analyze some examples of implementation for both families of *Darwin machines*. In Fig. 3.1, a graphical comparison between the theoretical Darwin Machine, the Genetic Darwin Machine, and Probabilistic Darwin Machine are shown.

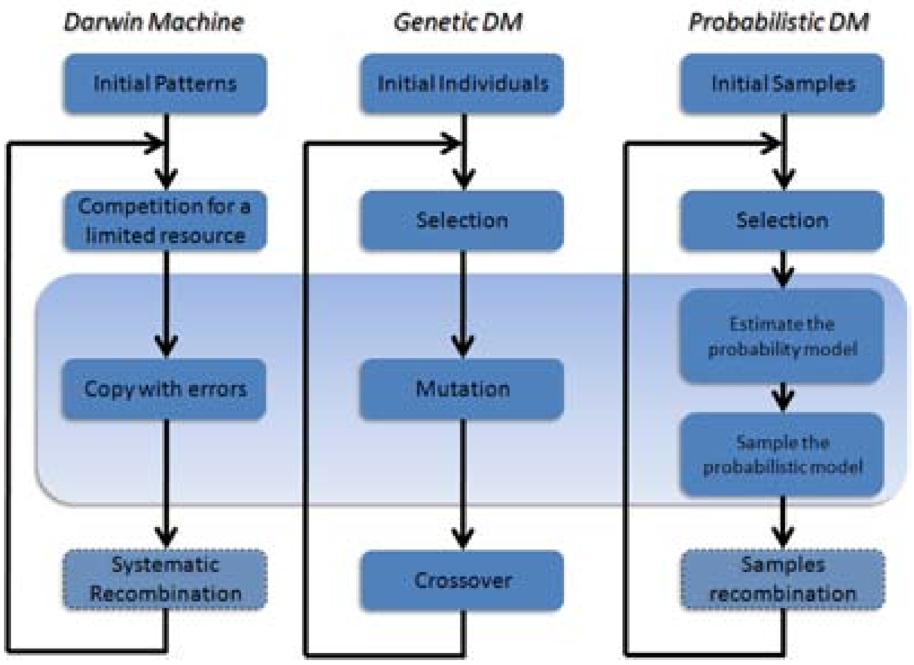


Figure 3.1: Graphical comparison between Darwin Machines, Genetic Darwin Machines and Probabilistic Darwin Machines. Although systematic recombination is an optional operation in Darwin Machines, it is usually implemented by Genetic Darwin Machines and can be also used on Probabilistic Darwin Machines.

3.1.1 Genetic Darwin Machine

All living organisms are *coded* by means of their genetic material, represented in DNA chains (see Fig. 3.2). The DNA contains all the information used in the development and functioning of all known living organisms and some viruses. Within cells, DNA is organized into structures called chromosomes, and the set of all the chromosomes are called genome. The genetic information is used as a receipt in order to create

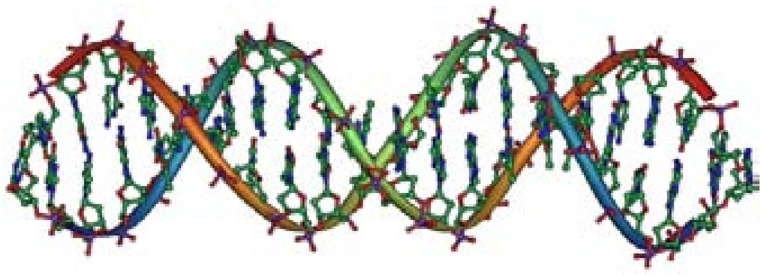


Figure 3.2: An schematic representation of the structure of part of a DNA double helix.

other structures of the cells as proteins, which are responsible of most part of the processes of the organism, and therefore, the way how the genetic information is expressed. Therefore, the capabilities and characteristics of an organism, and thus their adaption ability to the environment, depends on the information contained into the chromosomes. The evolution concerns to the natural processes that allows to perpetuate useful genetic information and improves it in order to adapt environment changes.

The basis of any evolutionary algorithms relies on the concept of *Natural Selection*, where the best individuals can survive and reproduce in order to perpetuate their specie. The genetic information of those best individuals is transferred among generations, getting better individuals and improving as specie. In general, in natural evolution, individuals are classified by their adaptation level: the best individuals are the most adapted to the environment. Apart from this information pathway through generations, there are two other sources of evolution:

Crossover: In the reproduction process, two individuals of a population interchange their genetic information, therefore, the offspring has a genetic information that contains parts of both their ancestors. When that mixing derives on a better adaption to the environment, the new individuals will survive and this new genetic information will persist generation after generation. In other case, that offspring will not survive, and the new information will be discarded.

Mutation: In contrast with crossover, where an existing information is combined, in the case of mutation, a new information can be generated. Mutation consists of randomly changing small parts of the genetic information. Analogously to crossover, when that changes implies a better adaptation, they will be passed generation after generation. Mutation allows evolution to generate new species.

Darwin explains the evolution of species (see Fig. 3.3) from the inference of those processes (selection and variability), and the posterior advances on genetics knowledge defined how these aspects are accomplished by all living beings in the nature. Biological evidences of the implementation of those concepts in the nature are used in order to simulate the natural evolution in a computational manner.

One of the first computational approaches to the natural evolution was presented by Nils A. Barricelli [Bar54], one of the pioneers in evolutionary computation. His initial experiments comprised a simple simulation of numbers in a grid. The numbers moved in the grid according to local rules that were specified for each number [Fog06]. Barricelli made several observations about the patterns that emerged from such simple rules, which he termed *organisms*. Organisms were defined to be independent if they could reproduce without requiring other organisms of a different pattern, which he described using the term *another species*. He noted patterns of recombination, including a multiple point operation in which two patterns would collide and the result would be a new self-sustaining pattern with numbers chosen from each of the *parents*. Overall, Barricelli's search for emergent patterns is reminiscent of the search for emergent properties in complex adaptive systems that pervaded artificial life research in the late 1980s and early 1990s.

Beginning on 1957 with the work [Fra57], the Australian quantitative geneticist Alex Fraser published a series of papers addressing different aspects from basic con-

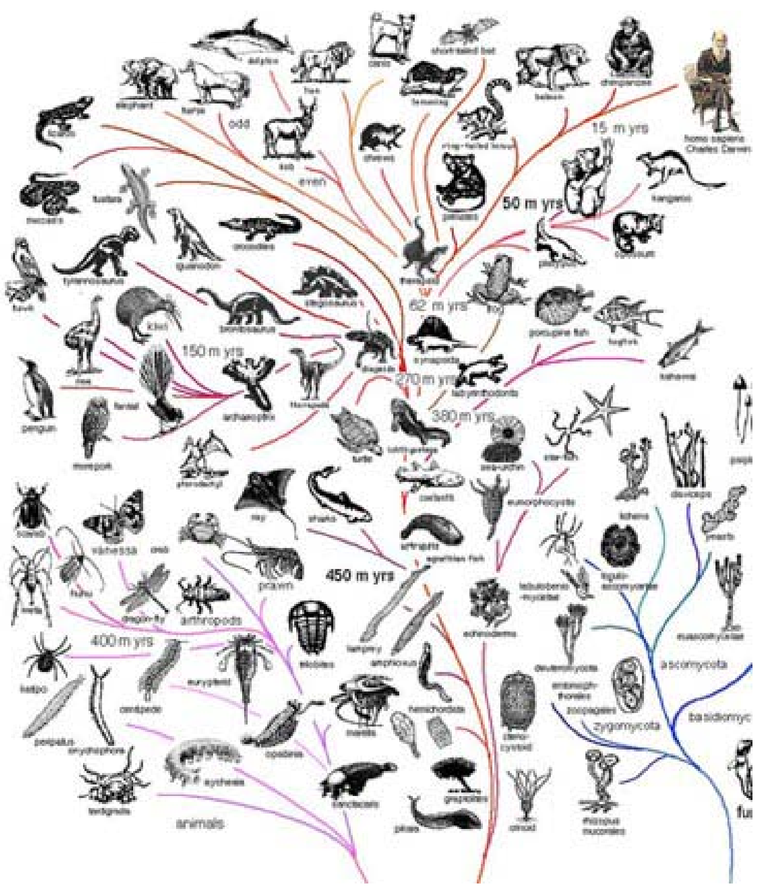


Figure 3.3: A schematic representation of the Darwin's evolution theory.

cepts to specific topics as the simulation of artificial selection of organisms with multiple loci (position of a gen into the genoma) controlling a measurable trait [FH65]. From these initial works, the basis of the evolutionary computation were established, and in early 1960s the computer simulation of evolution by biologists became more common and the methods were described in books by Fraser and Burnell [FB70], Crosby [Cro73], and the PhD thesis of Rechenberg [Rec71], which provided a comprehensive treatment of the different efforts spanning over a decade. Fraser's simulations included all of the essential elements of the most used and probably most known *Genetic Darwin Machine*, the *Genetic Algorithms*, but they did not become popular until the publication of applied works, such as the one of Schwefel [Sch81], where evolutionary computation was used to solve engineering problems.

From the popularization of evolutionary algorithms, they has been widely used in several optimization problems, such as scheduling and function minimization. The fundamental approach to optimization is to formulate a single standard of measurement (a cost function) that summarizes the performance or value of a decision and iteratively improve this performance by selecting from among the available alternatives. The cost function or evaluation function, is the measure of adaptation of a certain *organism* (solution) to the environment (problem). A deep study on the definition and implementation of *Genetic Algorithms* is presented in Section 3.2. Note that some common strategies commonly used on *Genetic Algorithms*, such as crossover and parallel evolutions are described in the general model as non essentials.

3.1.2 Probabilistic Darwin Machine

Apart from the evolutionary algorithms that simulate the natural behavior, recently, a new paradigm in the evolutionary computation field is taking huge relevance. This paradigm implements the Darwinian processes by means of probability models. The patterns are encoded by means of a set of random variables. These random variables are estimated by means of a probability model, which is sampled in order to obtain copies of the pattern with random variations. The fittest samples are used in order to estimate a new probability model (competition), which better represents the promising regions of the search space. Therefore, in this new paradigm, instead of working with a search space which points represent the patterns, in this case each point of the search space is a probability distribution, and the goal of the evolutionary process consist of finding the probability model which better represent the promising patterns of the original search space.

On of the most extended implementation for the Probabilistic Darwin Machines, is commonly known as *Evolutive Algorithms Based on Probabilistic Models* (EAPM) are a new paradigm which starts with the work of Baluja and Caruana [BC95], where the traditional operators of the genetic algorithms (mutation and crossover) had been replaced with the estimation and sampling of a probabilistic model. The new algorithm was named *Population Based Incremental Learning* (PBIL), and consists of a simple univariate model, where all the variables are assumed to be independent. The best individuals of each generation are used to update these variables, and finally the model is sampled to obtain a new generation. In spite of its simplicity, these algorithms demonstrated to converge to good solutions for several problems. Few years later, Schmidth et al. [SKJ99] re-introduced the genetic operators to the PBIL algorithm improving significantly its performance. In spite of this return to the origins, the PBIL algorithm introduced an interesting view on evolutionary computation: the extraction of a statistical description of the promising solutions, in terms of a probability distribution is the base of EAPMs and the new systematic way to solve hard search and optimization problems that they represent.

In the literature we can find a wide variety of EAPMs, in which the most important difference is the used probability model. Taking into account the considered interactions between variables, we can classify the models within three main types: Univariate models where no interactions are considered, bivariate models with only pair-wise interactions and finally the models that allow multiple interactions. Once the most convenient probability model is selected, different estimation and sample strategies can be used, thus, we can find different algorithms that share the same type of model. The most known and used algorithms are the UMDA [Müh97], PBIL [BC95] and cGA [HLG99] for univariate models, MIMIC [dBCIV97], COMIT [BD97b], and

BMDA [PM99] in the case of bivariate models and finally, considering models with multiple interactions the FDA [MM99], BOA [PGCP99], and EBNA [EL99].

The use of a complex model allows to better represent the features space, but it adds complexity to the estimation and sampling stages. In [BL01] a comparison between different EAPMs over several optimization problems suggests that for simple functions, where there are no interaction between the variables, the performance of the univariate and bivariate models perform as well as more complex models, but when we face more complex problems, a more sophisticated probability model is required. As a general rule, more complex models are more reliable but at the expense of bigger execution times.

3.2 Genetic Algorithms

Genetic Algorithms (GA) are the most common implementation for natural evolution simulation. The underlying idea is to reproduce the natural evolution by means of computer programs, using a chromosome based representation of the problems, and implementing from a functional point of view, the processes involved into the natural evolution. Genetic algorithms are often viewed as function optimizers, although the range of problems to which Genetic Algorithms have been applied is quite broad, and in general, they can be applied in all fields where optimization problems need to be faced. Using GA, we can optimize most type of parameters over practically any solution space, including continuous, discrete, combinatorial search spaces $\mathcal{Y}$ without and with constraints as well as mixed search spaces. Given the optimization problem

$$\mathbf{y}^* = \mathrm{argopt}_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{y}), \tag{3.1}$$

the *objective function* $f(\mathbf{y})$ to be optimized can be presented in mathematical form, via simulations, or even in terms of measurements obtained from real objects.

In a strict interpretation the genetic algorithm refers to a model introduced and investigated by John Holland [Hol75] and by students of Holland (such as DeJong [DeJ75]). It is still the case that most of the existing theory for genetic algorithms applies either solely or primarily to the model introduced by Holland as well as variations on what is commonly referred as the *canonical genetic algorithm.* In a broader usage of the term, a genetic algorithm is any population-based model that uses selection and recombination operators to generate new sample points in a search space $\mathcal{Y}$.

Usually there are only two main components of most Genetic Algorithms that are problem dependent: the problem encoding and the evaluation function. The rest of processes are standard operators that only depends on which type of encoding is used. In the following, these different parts and processes of a Genetic Algorithm are described in detail, and the most important definitions are given.

3.2.1 Problem Encoding

Problem encoding consists on the representation of a certain solution or point in the search space by means of a *genotype* [Hol75] or alternatively a *chromosome* [Sch87]. When the solutions or individuals are transformed in order to be represented in a chromosome, the original values (the individual) are referred as *phenotype*, and each one of the possible settings for a phenotype are the *alleles*.

Although the encoding depends on the problem we are solving, there are some standard approaches to define the chromosome, each one of them adapts better to some problem stereotype. The most common representation are:

Binary Encoding: Binary encoding is the most common, mainly because first works about GA used this type of encoding. In binary encoding every chromosome is a string of bits, 0 or 1. This encoding is often not natural for many problems and sometimes corrections must be made after crossover and/or mutation. Among possible bit-string representations, the Gray code is known to alleviate the "Hamming cliff" problem, therefore, is the common used binary representation. However, recent studies [CJ03b] argue that the improvement using this codification is limited to certain types of problems.

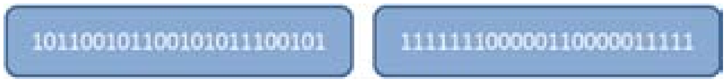


Figure 3.4: Codification using binary encoding.

Permutation Encoding: Permutation encoding can be used in ordering problems, such as travelling salesman problem or task ordering problem. In permutation encoding, every chromosome is a string of numbers, which represents number in a sequence. Permutation encoding is only useful for ordering problems.

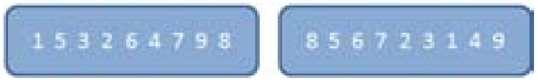


Figure 3.5: Codification using permuation encoding.

Value Encoding: Direct value encoding can be used in problems, where some complicated value, such as real numbers, are used. Use of binary encoding for this type of problems would be very difficult. In value encoding, every chromosome is a string of some values. Values can be anything connected to problem, form numbers, real numbers or chars to some complicated objects. Value encoding is very good for some special problems. On the other hand, for this encoding is often necessary to develop some new crossover and mutation specific for the problem.

Tree Encoding: Tree encoding is used mainly for evolving programs or expressions, for genetic programming. In tree encoding every chromosome is a tree of some objects, such as functions or commands in programming language. Tree encoding is good for evolving programs. Programming language LISP is often used to this, because programs in it are represented in this form and can be easily parsed as a tree, so the crossover and mutation can be done relatively easily.

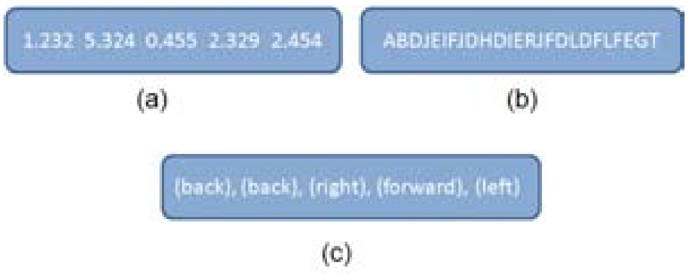

1.232 5.324 0.455 2.329 2.454
(a)
ABDJEIFJDHDIERJFDLDFLFEGT
(b)
(back), (back), (right), (forward), (left)
(c)

Figure 3.6: Codification using value encoding. *(a)* Real values *(b)* Char strings *(c)* Movements sequence.

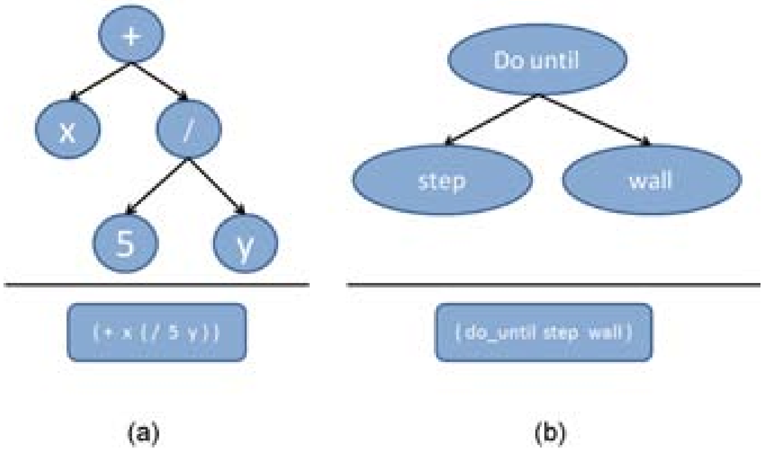


Figure 3.7: Tree encoding example. *(a)* Mathematical expressions. *(b)* Programming code.

3.2.2 Crossover

The crossover process consists on creating a new individual by means of the recombination of two individuals. Although crossover is one of the standard processes of a GA, most of the crossover strategies are designed for binary encoding. Other representations need special operations in order to guarantee that after crossover, the offsprings are consistent individuals. The idea behind crossover is that the new chromosome may be better than both of the parents if it takes the best characteristics from each of the parents. In the literature, we can find a set of predefined strategies:

n-point: A crossover operator that randomly selects n crossover points within a chromosome then interchanges the two parent chromosomes between these points to produce two new offspring. Common values for n are one and two.

Uniform: This operator decides with a certain probability (mixing ratio), which parent will contribute each of the gene values in the offspring chromosomes. This allows the parent chromosomes to be mixed at the gene level rather than the segment level (as with one and two point crossover). For some problems, this additional flexibility outweighs the disadvantage of destroying building blocks.

Arithmetic: In this case, the crossover operator linearly combines two parent chromosome vectors to produce two new offspring according to the following equations:

$$\begin{aligned}\text{Offspring}_1 &= a * \text{Parent}_1 + (1-a) * \text{Parent}_2\\ \text{Offspring}_2 &= (1-a) * \text{Parent}_1 + a * \text{Parent}_2\end{aligned}$$

where a is a random weighting factor (chosen before each crossover operation). In the case of binary encoding, arithmetic crossover can also be performed using standard logic operators, such as AND, OR, XOR...

Heuristic: In this case, the crossover operator uses the fitness values of the two parent chromosomes to determine the direction of the search. The offspring are created according to the following equations:

$$\begin{aligned}\text{Offspring}_1 &= \text{BestParent} + r * (\text{BestParent} - \text{WorstParent})\\ \text{Offspring}_2 &= \text{BestParent}\end{aligned}$$

where r is a random number between 0 and 1. It is possible that Offspring_1 will not be feasible. This can happen if r is chosen such that one or more of its genes fall outside of the allowable upper or lower bounds. For this reason, heuristic crossover has a user defined parameter n for the number of times to try and find an r that results in a feasible chromosome. If a feasible chromosome is not produced after n tries, the worst parent is returned as Offspring_1.

3.2.3 Mutation

Mutation consists of occasional changes in the value of certain positions of a chromosome. The motivation for using mutation is to prevent the permanent loss of any particular bit or allele. For instance, using a binary encoding, after several generations it is possible that selection will drive all the bits in some position to a single value, either 0 or 1. If this happens without the genetic algorithm converging to a satisfactory solution, then the algorithm has prematurely converged. This may particularly be a problem if one is working with a small population. Without a mutation operator, there is no possibility for reintroducing the missing bit value. The most common mutation operators:

Flip Bit: This operator simply inverts the value of the chosen gene (0 goes to 1 and 1 goes to 0). This mutation operator can only be used for binary genes.

Boundary: The value of a chosen gene is replaced with either the upper or lower bound for that gene (chosen randomly). This mutation operator can only be used for integer and float genes.

Non-Uniform: The probability that the amount of the mutation will be close to 0 increases with the generation number. This mutation operator keeps the population from stagnating in the early stages of the evolution then allows the genetic algorithm to fine tune the solution in the later stages of evolution. This mutation operator can only be used for integer and float genes.

Uniform: This operator replaces the value of the chosen gene with a uniform random value selected between the user-specified upper and lower bounds for that gene. This mutation operator can only be used for integer and float genes.

Gaussian: A mutation operator that adds a unit Gaussian distributed random value to the chosen gene. The new gene value is clipped if it falls outside of the user-specified lower or upper bounds for that gene. This mutation operator can only be used for integer and float genes.

Independently of the mutation strategy, it is necessary to guarantee a small mutation probability in order to avoid that the stochastic search becomes a random search.

3.2.4 Selection

Selection refers to the process analogous to the Natural Selection. The best adapted individuals will survive, while the rest disappear. The adaption level is measured by means of the *evaluation function* or *objective function*, a function which assigns to each individual a value reflecting their adaptation to the problem. The notion of evaluation and fitness are sometimes used interchangeably, however it is useful to distinguish between the evaluation function and the fitness function used by a Genetic Algorithm. The evaluation function provides a measure of performance with respect to a particular set of parameters, while the fitness function transforms that measure of performance into an allocation of reproductive opportunities. Therefore, the evaluation of a string representing a set of parameters is independent of the evaluation of any other string, however, the fitness of that string is always defined with respect to other members of the current population.

In the canonical genetic algorithm, fitness is defined by $f_i/\bar{f}$ where f_i is the evaluation associated with string i and $\bar{f}$ is the average evaluation of all the strings in the population. Fitness can also be assigned based on a string rank in the population [Bak85, Whi89] or by sampling methods such as tournament selection [Gol90]. In the following, some strategies are described:

Roulette Wheel: This is a way of choosing members from the population of chromosomes in a way that is proportional to their fitness. The method consists on allocating offspring strings using a roulette wheel with slots sized according to fitness. Parents are selected according to their fitness: the better the fitness of the chromosome, the greater the chance it will be selected. Main disadvantages of this approach is that the fittest member is not guaranteed to goes to the next generation, and if the difference between the fittest member and the rest is large, other members will have a slim chance of being selected. This selection method usually imply a faster convergence.

Rank: In order to give more chances to less fitted individuals, rank selection first ranks the population and then every chromosome receives fitness from this ranking. The worst will have fitness 1, second worst 2 etc. and the best will have fitness N, where N is the number of chromosomes in population. Although using this schemata the less fitted individuals have more chances to be selected, the convergence is usually slower than in the case of a fitness based roulette wheel approach.

Tournament: The most common tournament selection consists on randomly select n individuals from the population and store the fittest. The most common type of tournament selection is binary tournament selection, where just two individuals are selected.

Boltzmann: This is a method inspired by the technique of simulated annealing, selection pressure is slowly increased over evolutionary time to gradually focus the search. Given a fitness of f, Boltzmann selection assigns a new fitness, f_0, according to a differentiable function.

Apart from the selection methods described above, there are two possible designs for the selection process, the *Generational* one where the entire population is replaced at each generation, and the *Steady-state* approach, where only part of the population is replaced at each generation. The major difference between steady-state and generational approaches is that for each N members of the population generated in the generational approach, there are $2 \times N$ selections. Consequently the selection strength and genetic drift for a steady-state GA is twice than in the generational GA. The steady-state GA, therefore, appears twice as fast although it can lose out in the long term because it does not explore the landscape as well as the generational GA.

A part from the above standard methodologies, another criterions can be incorporated to the selection phase. For instance, the diversity of individuals in a population is often desirable, so we can introduce diversity measures on the fitness function, selecting not only good fitted individuals, but different individuals.

Since the selection process is based on probabilistic methodologies, the survivor of the fittest individual of a population is not guaranteed. In order to never loose the best solutions found during the evolution process, it is common to apply an *elitist criterium*, which selects the fittest n individuals to pass directly to the next generation. Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution to date.

3.2.5 Evolution

Once all the main definitions and processes for a Genetic Algorithm are given, it is the moment to put all them together. The algorithm for a Genetic Algorithm is shown in Algorithm 7 and Fig. 3.8. The processes described above are iteratively repeated in order to evolve an initial group of individuals to a better adapted individuals. The stopping condition are defined over problem knowledge and general convergence criterions. If the goal value is known, the evolution process can be stopped when the better individual of the population achieves this value. Otherwise, we can stop using convergence criterions as the best individual value progression or the divergence on the population.

Algorithm 7 The canonical Genetic Algorithm

```
Input: Evaluation function $f(x)$ and the user defined parameters (mutation rate,
  crossover, etc...)
  Generate a random initial population $P_0$
  repeat
    Use the evaluation function to determine the fitness of $P_t$
    Select parents from population $P_t$
    Perform crossover on parents creating population $P_{t+1}$
    Perform mutation of population $P_{t+1}$
  until Best individual is good enough or some stopping criteria is met.
Output: The best individual of the last population.
```

The parameters of a GA must be chosen for each problem, and it is often performed by means of some type of cross-validation method. In general, it is important to control the number of evaluations, which is the main responsible of the learning cost. Concerning to the crossover and mutation operators, in general, it is good to have both operators in the evolution process, although mutation only GA are possible. In contrast, crossover only GA would generally not work.

In the natural evolution, individuals of a population creates sub-populations, mainly limited to a certain geographic area or either social classes. It means that parallel evolution processes are performed with a small number of individuals that can migrate from one region to another one. An step beyond in the GAs theory, is the inclusion of that type of parallelization.

3.2.6 Parallelization

Part of the biological metaphor used to motivate genetic search is that it is inherently parallel. In natural populations, thousands or even millions of individuals exist in parallel. This suggests a degree of parallelism that is directly proportional to the population size used in genetic search. In the following, three different ways of exploiting parallelism in genetic algorithms are described:

Global Population: The most direct way to implement a parallel genetic algorithm is to implement something close to a canonical genetic algorithm. The only change that will be made is that selection will be done by Tournament Selection. First, selection is used to create an intermediate population of duplicate strings selected according to fitness and then crossover and mutation are applied to produce the next generation.

Island Model: The motivation for using Island Models is to exploit a more coarse grain parallel model. The main idea is to split a population of N individuals into S sub-populations of N/S individuals. If each sub-population is evolved independently from the others, genetic drift will tend to drive these populations in different directions. By introducing migration, the Island Model is able to

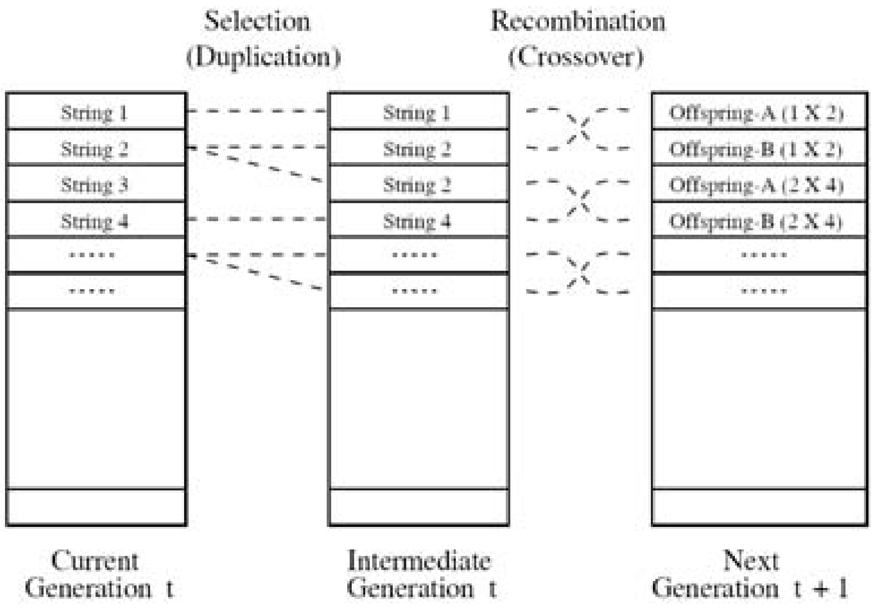


Figure 3.8: One generation is broken down into a selection phase and recombination phase. This figure shows strings being assigned into adjacent slots during selection. In fact they can be assigned slots randomly in order to shuffle the intermediate population. Mutation not shown can be applied after crossover [Whi94].

exploit differences in the various sub-populations (this variation in fact represents a source of genetic diversity). Each sub-population is an island and there is some designated way in which genetic material is moved from one island to another.

Cellular: This schemata assumes a grid of cells (or processors), each one with its own sub-population. The effects of evolving these sub-populations are the same that in the case of an Island Model, but now, migrations are restricted to the local neighborhoods. Each processor can pick the best string in its local neighborhood to mate with or alternatively some form of local probabilistic selection could be used. In either case, only one offspring is produced and becomes the new resident at that processor.

3.3 Evolutionary Algorithms Based on Probabilistic Models

3.3.1 Population Based Incremental Learning

The PBIL algorithm as defined by Baluja [BC95, BD97b] is a combination of evolutionary optimization and competitive learning. The goal of this algorithm is to create a real valued probability vector which, when sampled, reveals high evaluation solution vectors with high probability. In other words, it explicitly maintains statistics about the search space and uses them to direct its exploration.

In [BC95], Baluja uses a binary encoding for the chromosomes, therefore, all the values for the initial distribution are initialized to 0.5. As the search proceeds, the values in the probability vector gradually shift to represent high evaluation vectors. This is accomplished as follows (see Algorithm 8): A number of solution vectors are generated based upon the probabilities specified in the probability vector. The probability vector is pushed towards the generated solution vectors with the highest evaluation values. After the probability vector is updated, a new set of solution vectors is produced by sampling from the updated probability vector. After some iterations, if the algorithm converges all the values in the probability vector will be near 1 or 0.

Algorithm 8 The *Population Based Incremental Learning* Algorithm

```
Input: Evaluation function $f(\mathbf{x})$, the chromosome length $L$, the number of samples $M$, the learning rate $\alpha$, and the number of vectors to update from $S < M$
  Initialize the probability vector $P = \{p_1, p_2, \dots, p_L\}$ with $\forall i = 1, ..., L | p_i = 0.5$
  while The termination condition is not met do
    for $s = 1, ..., M$ do
      Generate a solution vectors $V[i]$ according to probabilities $P$
      $E[i] \leftarrow f(V[i])$
    end for
    Create a sorted copy $V^\star$ of vectors in $V$ according to their evaluation values $E$
    for $j = 1, \dots, S$ do
      $\mathbf{v} \leftarrow V^\star[j]$
      for $i = 1, \dots, L$ do
        $p_i \leftarrow p_i * (1.0 - \alpha) + \mathbf{v}_i * \alpha$
      end for
    end for
  end while
Output: The fittest vector found during the iterations, or the probability vector $P$ if the algorithm converges.
```

Although it simplicity, Baluja experiments demonstrated that this algorithm performs as well as a Genetic Algorithm. In [BC95], some variations of the basic PBIL algorithm that can improve search effectiveness:

Mutations: Similarly, when the probability vector in PBIL converges towards 0s and 1s, exploration also is reduced. Mutations perturb the probability vector with a small probability in a random direction. The amount of the perturbation is generally kept small in relation to the learning rate.

Negative Samples: A second variation is to learn from negative examples instead of only positive ones. In the PBIL algorithm, the probability vector is updated towards the N_U best vectors in the population. However, the probability vector can also be shifted away from the worst vectors.

The work of Baluja and Caruana [BC95] was presented as a way to remove mutations and crossover operators from the standard Genetic Algorithms, replacing them by the estimation of probability models and posterior sampling of these models (see Fig. 3.1). Later, Schidth [SKJ99] reintroduced those operators to the PBIL schemata, but the underlying idea of representing the knowledge acquired during the evolution process by means of probability models, opened a new field referred as *Evolutionary Algorithms based on Probabilistic Models* (EAPM). Although this is the nomenclature we will adopt in this thesis, this family of algorithms can be found in the literature under different names, such as *Estimation of Distributions Algorithms*, *Iterated Density Estimation Algorithms*, or *Probabilistic Model-Building.*

The transition from Genetic Algorithms to EAPM is performed by assuming each gene of the chromosome as a random variable, that is, a numerical outcome of a random experiment. Therefore, a chromosome can be seen as a vector of random variables and our goal is to learn the behavior of these values. More precisely, we will estimate a probability model over these random variables for the promising regions of the search space.

In the case of PBIL, interactions between the different genes are not considered, therefore, the probability model corresponds to an univariate model. In addition, in the presented case, chromosomes are encoded using binary codification, and thus, the random variables are discrete and limited to only two values. If we consider interaction or dependence between different random variables in a chromosome, and non binary encodings, the probability models for the random variables is considerable more complex, which in general needs more complex methods to be estimated.

In the rest of this section, different EAPM's are introduced, grouped by the complexity on the base probability models. More detailed studies on EAPM with deeper analysis and wide comparisons can be found in the works of Larrañaga et al. [LL02, LLM03, LLIB06].

3.3.2 EAPM based on univariate models

The univariate models assume that there is no relation between the different random variables (see Fig. 3.9). In general, these algorithms approximate the $n-$dimensional join probability distribution $p(\mathbf{x})$ as a product of n independent univariate probability distributions, that is, $p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i)$. This assumption in general is not true, in special when an optimization problem is faced.

The most representative algorithms in this category are the *Population Based Incremental Learning* (PBIL), the *Univariate Marginal Distribution Algorithm*(UMDA), and the *compact Genetic Algorithm* (cGA). These algorithms are introduced in the following.

UMDA

Univariate Marginal Distribution Algorithm was introduced by Mühlenbein in [Müh97]. In order to represent the promising regions of the search space, the join probability distribution is estimated from a set $D^\star$ of S selected individuals by means of a product of independent univariate distributions. In this case, the univariate distributions

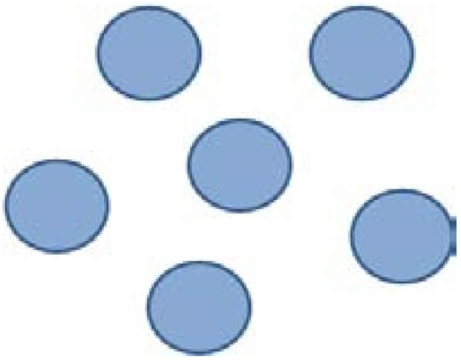


Figure 3.9: Graphical representation for the univariate probabilistic models in EAPM. No interactions are considered between variables.

are estimated as:

$$p(x_i) = \frac{\sum_{j=1}^{S} \delta_j(X_i = x_i|D^\star)}{S} \tag{3.2}$$

where

$$\delta_j(X_i = x_i|D^\star) = \begin{cases} 1 & \text{if the } j\text{−th sample of } D^\star,\ X_i = x_i \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

The pseudo-code for the UMDA algorithm is shown in Algorithm 9. Although is out of the scope of this thesis, as in the case of PBIL, in the literature we can find different modification for the UMDA algorithm.

Algorithm 9 The *Univariate Marginal Distribution Algorithm*

```
Input: Evaluation function $f(\mathbf{x})$, the chromosome length $L$, the number of samples
  $M$, and the number of vectors to update from $S < M$
  Generate an initial random population $D_0$ with $M$ individuals.
  $l \leftarrow 0$
  while The termination condition is not met do
    Evaluate the population $D_l$ using the evaluation function $f$.
    Select the fittest $S$ individuals of $D_l$ $(D_l^\star)$
    Estimate $p_l(\mathbf{x}) = p_l(\mathbf{x}|D_l^\star) = \prod_{i=1}^{L} p_l(x_i) = \prod_{i=1}^{L} \frac{\sum_{j=1}^{S} \delta_j(X_i=x_i|D_l^\star)}{S}$
    Sample $M$ individuals using $p_l(\mathbf{x}$
  end while
Output: The fittest vector found during the iterations.
```

cGA

The *compact Genetic Algorithm* was introduced by Harik et al. [HLG99]. This algorithm, in their binary form, initialize a probability vector with a Bernoulli distribution with parameter $p = 0.5$. At each iteration, the probability model is sampled in order to obtain two individuals. Once the individuals are evaluated, the positions for which the individuals have different values are updated, increasing the probability for the value in the best adapted individual. This update of the probability model towards the winning individual is repeated until the parameters of all Bernoulli distributions corresponds to 0 or 1 (the algorithms converges). Once converged, the final probabilities are considered the result of the process. The Pseudo-code for this algorithm is shown in Algorithm 10.

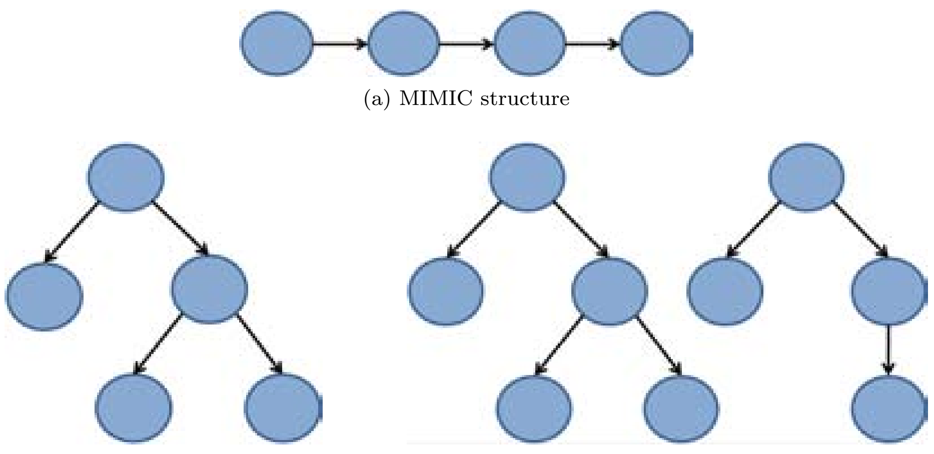
Algorithm 10 The *compact Genetic Algorithm*

```
Input: Evaluation function $f(\mathbf{x})$, the chromosome length $L$, and the order of the linking block $k$.
  Initialize the probability vector $P_0(\mathbf{x}) = P_0(x_1, \dots, x_L) = (p_0(x_1), \dots, p_0(x_L))$ with $\forall i = 1, ..., L | p_0(x_i) = 0.5$
  $j \leftarrow 0$
  while $\exists i | p_j(x_i) > 0$ or $p_j(x_i) < 1$ do
    $j \leftarrow j + 1$
    Sample $P_j$ in order to obtain two individuals $x^1$ and $x^2$.
    Evaluate $x^1$ and $x^2$ using the evaluation function $f$.
    Sort the individuals in a manner that $x^{1\star}$ is the best and $x^{2\star}$ the worse.
    Move the probability vector towards $x^{1\star}$:
    for $i = 1, \dots, L$ do
      if $x_i^{1\star} \neq x_i^{2\star}$ then
        $$p_j(x_i) = \begin{cases} p_{j-1}(x_i) - \frac{1}{k} & \textbf{if } x_i^{1\star} = 0 \\ p_{j-1}(x_i) + \frac{1}{k} & \textbf{if } x_i^{1\star} = 1 \end{cases}$$
      end if
    end for
  end while
Output: The probability vector $P_j(\mathbf{x})$
```

3.3.3 EAPM based on bivariate models

Univariate models offers a simple way to estimate a join probability distribution, at the cost of assuming no relations between variables. Considering pair-wise relations, the join probability distribution can be also estimated in a easy manner, with a weaker assumption. In this case, statistics of second order are calculated, and in contrast with the previous methods, where only the parameters were estimated, in this case we need to estimate both, parameters and structure.

Some of most common algorithms based on bivariate models are presented in the following, and in Fig. 3.10, a graphical representation of their probability models is shown. Note, that there are different strategies in order to represent the pair-wise relations.


(a) MIMIC structure

(b) Tree structure

(c) BMDA approach

Figure 3.10: Graphical representation for the bivariate probabilistic models in EAPM with pair-wise relations between variables.

MIMIC

The *Mutual Information Maximization for Input Clustering* algorithm was introduced by De Bonet et al. in [dBCIV97]. This algorithm search the best permutation of variables in order to find a probability distribution $P^{\pi}(\mathbf{x})$ that minimizes the Kullback-Leibler divergence with the empiric distribution for a set of selected individuals:

$$P^{\pi}(\mathbf{x}) = P(x_{i_1}|x_{i_2}) \times \ldots \times P(x_{i_{n-1}}|x_{i_n}) \times P(x_{i_n}) \tag{3.4}$$

where $\pi = (i_1, \ldots, i_n)$ denotes a permutation of the indexes $1, 2, \ldots, n$.

It can be demonstrated, that the Kullback-Divergence between two probability distributions $P_1(\mathbf{x}$ and $P_2(\mathbf{x}$, can be written as:

$$H^{\pi}(\mathbf{x}) = h(X_{i_n}) + \sum_{j=1}^{n-1} h(X_{i_j}|X_{i_{j+1}}) \tag{3.5}$$

where $h(X) = -\sum_x P(X = x) \log P(X = x)$ is the Shannon entropy for the variable X, and $h(X|Y) = \sum_y h(X|Y = y)p(Y = y)$ with $h(X|Y = y) = -\sum_x P(X = x|Y = y) \log P(X = x|Y = y)$ as the conditional entropy for X given Y. Therefore, the problem of finding the best permutation $P^{\pi}(\mathbf{x})$ is equivalent to search the permutation $\pi^{\star}$ which minimizes $H^{\pi}(\mathbf{x})$.

The approach of De Bonet et al. [dBCIV97] is the following: Given some cost function $f(\mathbf{x})$ with local minima, knowing nothing else about $f(x)$ it might not be unreasonable to search for its minimum by generating points from a uniform distribution over the inputs $P(\mathbf{x})$. Such a search allows none of the information generated by previous samples to effect the generation of subsequent samples. Not surprisingly, much less work might be necessary if samples were generated from a distribution, $P^{\theta}(\mathbf{x})$, that is uniformly distributed over those $\mathbf{x}$'s where $f(\mathbf{x}) \leq \theta$ and has a probability of 0 elsewhere, being θ the median fitness value. For example, if we had access to $P^{\theta}(\mathbf{x})$ for $\theta = min_x f(\mathbf{x})$ a single sample would be sufficient to find an optimum. Using this insight, given a collection of points for which $f(\mathbf{x}) \leq \theta_0$ a density estimator for $P^{\theta_0}(\mathbf{x})$ is constructed. From this density estimator, additional samples are generated, a new threshold value $\theta_1 = \theta_0 - \epsilon$ is established, and a new density estimator $P^{\theta_1}(\mathbf{x})$ is constructed. This process is repeated until the value of $f(\mathbf{x})$ cease to improve. This process is summarized in Algorithm 11.

Algorithm 11 The *Mutual Information Maximization for Input Clustering* Algorithm

```
Input: Evaluation function $f(\mathbf{x})$, the number $M$ of samples, and the $N$th-percentile
  to set the threshold.
  Generate an initial random population $D_0$ with $M$ individuals.
  Evaluate the individuals in $D_0$ using the evaluation function $f$.
  Set $\theta_0$ as the median fitness value of $D_0$.
  $j \leftarrow 0$
  repeat
    Select $i_n = \text{argmin}_i \hat{h}_j(X_i)$, where $\hat{h}(X)$ is the empiric entropy of $X$.
    for $k = n-1, \ldots, 1$ do
      Select $i_k = \text{argmin}_l \hat{h}_j(X_l|X_{i_{k+1}})j \neq i_{k+1}, \ldots, ..., i_n$, where $\hat{h}(X|Y)$ is the empiric conditional entropy of $X$ given $Y$.
      $P_j^{\pi}(\mathbf{x}) = P_j(x_{i_1}|x_{i_2}) \times \ldots \times P(x_{i_{n-1}}|x_{i_n}) \times P(x_{i_n})$
    end for
    Generate more samples from the distribution $P^{\theta_j}(\mathbf{x})$.
    Set $\theta_{j+1}$ equal to the $N$th percentile of the data.
    Select the samples which $f(\mathbf{x}) < \theta_{j+1}$.
    $j \leftarrow j+1$
  until $(\theta_{j-1} - \theta_j) \leq 0$
Output: Best individual found during the process.
```

COMIT

The *Combining Optimizers with Mutual Information Trees* algorithm was proposed by Baluja and Davies in [BD97b]. The authors use bivariate probability models in order to find an optimal likelihood dependency tree. This method consists on a hybrid approach that combines the EAPM approach with a local optimizer.

Following a similar approach that in the case of MIMIC, the goal is to model a probability distribution $P(X_1, \ldots, X_n)$ over bit-strings of length n, where $X_1, \ldots, X_n$ are variables corresponding to the values of the bits. It is supposed that the model $P'(X_1, \ldots, X_n)$is restricted to models of the following form:

$$P'(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_{m(i)}|X_{m(p(i))}) \tag{3.6}$$

where $m = (m_1, \ldots, m_n)$ is some unknown permutation of $(1, \ldots, n)$, and $p(i)$ maps the integers $0 < i \leq n$ to integers $0 \leq p(i) < i$. $P(X_i|X_0)$ is by definition equal to $P(X_i)$ for all i. In other words, P' is restricted to factorizations in which the conditional probability distribution for any one bit depends on the value of at most one other bit, or in Bayesian network terms, it is restricted to networks in which each node can have at most one parent.

In order to find the optimal model within these restrictions, Baluja and Davies use the *Maximum Weight Spanning Tree* (MWST), a method proposed by Chow and Liu in [CL68]. In this method, a complete weighted graph G is created in which every variable X_i is represented by a corresponding vertex V_i, and in which the weight W_{ij} for the edge between vertices V_i and V_j is set to the mutual information $I(X_i, X_j)$ between X_i and X_j. The edges in the maximum spanning tree of G determine an optimal set of $(n-1)$ first-order conditional probabilities with which to model the original probability distribution. Since the edges in G are undirected, a decision must be made about the directionality of the dependencies with which to construct P'. However, all such orderings conforming to Equation 3.6 model identical distributions. Among all trees, this algorithm produces the tree which maximizes the likelihood of the data when the algorithm is applied to empirical observations drawn from any unknown distribution.

Baluja and Davies use the MWST algorithm for combinatorial optimization as follows: They incrementally learn second-order statistics from previously seen *good* individuals. Then, using the MWST algorithm, they determine optimal subsets of these statistics with which to create model probability distributions $P'(X_1, \ldots, X_n)$ of the form assumed in Equation 3.6. These distributions are used to generate new candidate individuals which are then evaluated. The best individuals are used to update the second-order statistics. Finally, these statistics are used to generate another dependency tree and this process is repeated until the algorithm's termination criteria are met. The pseudo-code for this algorithms is shown in Algorithm 12.

Algorithm 12 The *Combining Optimizers with Mutual Information Trees* Algorithm

```
Input: Evaluation function $f(\mathbf{x})$, the number $M$ of samples, and the number of
  individuals $S < M$ to select.
  Generate an initial random population $D_0$ with $M$ individuals.
  Evaluate the individuals in $D_0$ using the evaluation function $f$.
  $j \leftarrow 0$
  while Stopping criteria is not met do
    Select a subpopulation $D_j^\star$ with $M$ individuals from $D_j$ according their evaluation
    value $f(\mathbf{x})$.
    Use the MWST algorithm [CL68] to estimate $P_l'(\mathbf{x}) = P'(\mathbf{x}|D_j^\star) = \prod_{i=1}^{n} P_j(x_i|x_{p(i)})$
    Sample a new population $D_j'$ of $M$ individuals from $P'(\mathbf{x})$.
    Select a subpopulation $D_j^{F-S}$ of $M - N$ individuals by means of a fast method,
    beginning with the best individual in $D_j'$.
    Create the next population as: $D_j + 1 = D_j^\star \cup D_j^{F-S}$
    $j \leftarrow j + 1$
  end while
Output: Best individual found during the process.
```

BMDA

Pelikan and Mühlenbein [PM99] introduced the *Bivariate Marginal Distribution Algorithm*, an extension for the UMDA which uses a factorization for the joint probability distribution that only requires statistics of order two. This method is based on the creation of a directed graph that represents the dependencies. This graph cannot has cycles and can has unconnected nodes. This model can be seen in terms of graphical models, as a set of trees.

Before presenting the algorithm, we need to define the Pearson χ^2 statistic, used in order to evaluate the dependency between different variables. The Pearson χ^2 statistic can be written as [Pla83]:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{3.7}$$

where O_i is the observed value and E_i the expected value. For each pair of positions, the observed quantity is the number of possible pairs of values on these positions. If these two positions are independent, the number for each of these pair of values could be easily calculated using the basic probability theory. This is the expected quantity. Then, in terms of univariate and bivariate frequencies and the total number of points N taken into account, for positions $i \neq j$, we get:

$$\chi^2_{i,j} = \sum_{x_i,x_j} \frac{\left(N \times P_{i,j}(x_i,x_j) - N \times P_i(x_i)P_j(x_j)\right)^2}{N \times P_i(x_i)P_j(x_j)} \tag{3.8}$$

where $P_i(x_i)$ corresponds to the univariate marginal probability defined as the frequency of individuals that have x_i on the ith position, and $P_{i,j}(x_i,x_j)$ is the bivariate marginal probability defined as the frequency of individuals that have x_i and x_j on positions i and j respectively. If positions i and j are statistically independent with a confidence value of 95%, then for Pearson's χ^2 statistics, the following inequality holds:

$$\chi^2_{i,j} < 3.84 \tag{3.9}$$

In order to build those graphs, the authors select an arbitrary variable and add it to the graph. Then, using the Pearson χ^2 statistic from Equation 3.8, the most dependant variable to the previous one is added. This process is repeated, adding the variables in the set of not yet added variables that are most dependant to one of the set of added variables. If in any moment, all the dependence values between selected variables and not added variables are lower than the minimum dependency value defined in Equation 3.9, this process is stopped. Now we randomly select one of the non selected variables and repeat the above process until all the variables are in the graph. This process is summarized in Algorithm 13.

The factorization of the probability model at each iteration l can be written as:

$$P_l(\mathbf{x}) = \prod_{X_r \in R_l} P_l(x_r) \prod_{X_i \in V \setminus R_l} P_l(x_i|x_{j(i)}) \tag{3.10}$$

Algorithm 13 Algorithm for the Construction of a Dependency Graph in BMDA.

```
Input: Univariate $P_i$ and bivariate $P_{i,j}$ frequencies for each variable.
  Set $V \leftarrow \{0, \dots, n-1\}$, the set of vertices. $i$ corresponds to the $i$th position of the individual.
  Set $A \leftarrow V$, the set of not yet processed vertices.
  Set $E \leftarrow \emptyset$, the set of edges.
  Label 1:
  Select an arbitrary vertex $v$ from $A$.
  Add $v$ into $R$, a set containing the root node of each connected component.
  Label 2:
  Remove $v$ from $A$
  if There are no more vertices in set $A$ then
    Goto Label 3
  end if
  if There are no more dependencies in $D$ of any $v \in A$ and $v' \in V \setminus A$ then
    Goto Label 1
  end if
  Set $v$ to the vertex from $A$ that maximizes $\chi^2_{v,v'}$ over all $v \in A$ and $v' \in V \setminus A$
  Add edge $(v, v')$ into the set of edges $E$.
  Goto Label 2
  Label 3:
Output: The Graph $G = (V, E, R)$.
```

where V is the set of n variables, R is the set that contains the root variables for each connected component (at iteration l), and $X_{j(i)}$ represents the variable connected to X_i and added before than X_i.

The probabilities on the root nodes, $P_l(x_r)$, as in the case of the conditional probabilities $P(x_i|x_{j(i)})$, are estimated from the selected individuals $D^{\star}_{l-1}$. In the Algorithm 14, the pseudo-code for the BMDA is shown.

3.3.4 EAPM based on multiple dependencies models

After the overview of EAPMs based on univariate and bivariate probability models, a further step is to consider multiple dependencies over the random variables. Most of the works that consider multiple dependencies are based on Bayesian networks to codify the probability distributions. In the literature, we can find EAPMs that use a join probability distribution factorization based on statistics with order greater than two. Despite the lack of implementation evidences, one of the first works to consider multiple dependencies in the EAPM framework was the one of Baluja and Davies [BD97a]. The most widely applied methods for multiple dependencies are introduced in the following, and a graphical representation of their probability models is shown in Fig. 3.11.

Algorithm 14 The *Bivariate Marginal Distribution Algorithm*

```
Input: Evaluation function $f(\mathbf{x})$.
  Set $l \leftarrow 0$
  Generate a random initial population $D_0$.
  while Termination criteria are not met do
    Select parents $D_l^\star$ from $D_j$ according their evaluation value $f(\mathbf{x})$.
    Calculate univariate frequencies $P_i$ and bivariate frequencies $P_{i,j}$ for $D_l^\star$
    Create dependency graph $G = (V, E, R)$ using the frequencies $P_i$ and $P_{i,j}$ and
    Algorithm 13.
    Generate the set of new individuals $O_l$ using dependency graph $G$ and frequencies
    $P_i$ and $P_{i,j}$.
    Replace some of individuals from $D_l$ with new individuals $O_l$
    $l \leftarrow l + 1$
  end while
Output: Best individual found during the process.
```

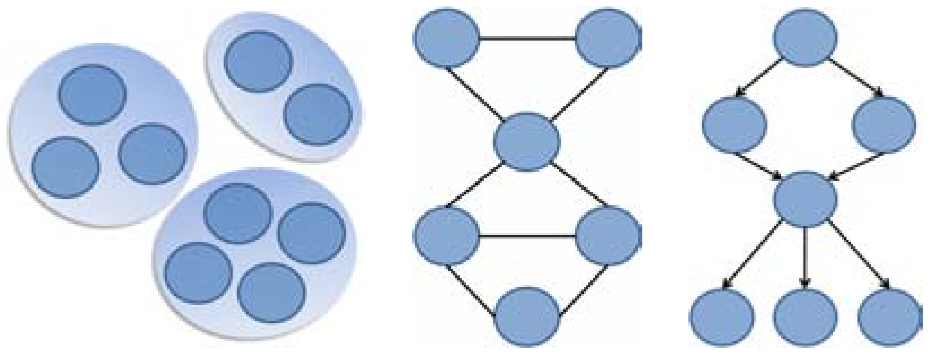

(a) EcGA structure (b) FDA structure (c) EBNA and BOA structure

Figure 3.11: Graphical representation for the probabilistic models in EAPM considering multiple relations between variables.

EcGA

The *Extended compact Genetic Algorithm* was introduced by Harik in [Har99]. The underlying idea is to use a factorization of the join probability model as a variable length product of marginal distributions. They use a rapid method in order to find groups of related variables, where each group is considered independent from the rest of the groups. The length of each product is related to the number of variables in the same group. Using this method, the join probability distribution of the n variables is calculated as:

$$P(\mathbf{x}) = \prod_{c \in C} P(\mathbf{x}_c) \tag{3.11}$$

where C is the set of groups, and $P(\mathbf{x}_c)$ is the marginal distribution of the variables on group c. Since this algorithm builds disjoint groups of variables, for all $c, k \in C$ we can assume:

$$\bigcup_{c \in C_l} \mathbf{X}_c = \{X_1, \ldots, X_n\}, \mathbf{X}_c \cap \mathbf{X}_k = \emptyset \tag{3.12}$$

In order to create the groups of variables, Harik initially builds an initial partition of n groups of one variable. From this starting point, this algorithm begins an iterative process that fuse pairs of groups. In order to select the groups to be fused, the author defines the *combined complexity*, a measure based in a combination of the sum of the marginal distributions entropies and a complexity penalization based on the *minimum description length principle* [Ris78], a formalization of the *Occam's Razor.* At each iteration, those two groups that being fussed obtains the higher reduction of this measure are selected in order to be fused, creating a larger group with the variables contained in both of them.

The *combined complexity* measure can be defined as the sum of two complexities, the population complexity

$$J_p = N \sum_{c \in C} h(\mathbf{X}_c = \mathbf{x}_c) = -N \sum_{c \in C} \sum_{\mathbf{x}_c} P(\mathbf{X}_c = \mathbf{x}_c) \log P(\mathbf{X}_c = \mathbf{x}_c) \tag{3.13}$$

and the model complexity

$$J_m = \log N \sum_{c \in C} dim \mathbf{X}_c \tag{3.14}$$

where $dim\mathbf{X}_c$ is the number of required parameters in order to describe the marginal distribution $\mathbf{X}_c$. For instance, in the case that all the variables in the c-th group were binary, $dim\mathbf{X}_c = 2\mathbf{X}_c| - 1$. In Algorithm 15 the learning process is detailed.

FDA

The *Factorized Distribution Algorithm* was introduced by Mühlenbein and Mahnig in [MM99]. The use of FDA requires to be applied over *additively decomposed functions* (ADF), that is:

Definition 8 *An additively decomposed function is defined by*

$$f(\mathbf{x}) = \sum_{s_i \in S} f_i(\Pi_{s_i} \mathbf{x}) \quad S = \{s_1, \ldots, s_l\}, s_i \subseteq \tilde{X} \tag{3.15}$$

where

$$\begin{aligned} \tilde{X} &= \{x_1, \ldots, x_n\} \quad \mathbf{B} = \{0,1\} \quad X = \mathbf{B}^{|\tilde{X}|} \\ X_s &\subseteq X \mathit{with} s \subseteq \tilde{X} \\ \Pi_s x &= \mathit{the\ projection\ of}\ x \in X\ \mathit{onto\ the\ subspace}\ X_s \end{aligned}$$

Over the ADF, the authors propose to use a generalization of the Boltzmann distribution in order to generate promising points:

Algorithm 15 The *Extended compact Genetic Algorithm*

```
Input: Evaluation function $f(\mathbf{x})$.
  Set $l \leftarrow 0$
  Generate a random initial population $D_0$ with $M$ individuals.
  while Termination criteria are not met do
    Select a subpopulation $D_l^\star$ with $S < M$ individuals from $D_j$ using tournament
    selection with the evaluation function $f(\mathbf{x})$.
    Calculate univariate frequencies $P_i$ and bivariate frequencies $P_{i,j}$ for $D_l^\star$
    Build the best clustering of variables $C_l$ in order to minimize:
    $J_l = J_p + J_m = \left(-N \sum_{c \in C_l} \sum_{\mathbf{x}_c} P(\mathbf{X}_c = \mathbf{x}_c) \log P(\mathbf{X}_c = \mathbf{x}_c)\right) + \left(\log N \sum_{c \in C_l} dim \mathbf{X}_c\right)$
    Calculate $P_l(\mathbf{x}) = P(\mathbf{x}|D_l^\star) = \prod_{c \in C_l} P_l(\mathbf{x}_c|D_l^\star)$
    Sample a new population $D_{l+1}$ with $M$ individuals from $P_l(\mathbf{x})$
    $l \leftarrow l + 1$
  end while
Output: Best individual found during the process.
```

Definition 9 *The Boltzman distribution of a function f is defined for $u \geq 1$ by*

$$P(\mathbf{x}) = \frac{u^{f(\mathbf{x})}}{\sum_y u^{f(y)}} \tag{3.16}$$

The Boltzmann distribution has the following feature: the larger the function value $f(\mathbf{x})$ becomes, the larger $P(\mathbf{x})$ becomes (for $u \geq l$). Although Boltzmann distribution is a good method to distribute the points, unfortunately, its computation needs an exponential effort (in the size of the problem). The distribution is factored into a product of marginal and conditional probabilities, defined as:

$$P(\Pi_{c_i}\mathbf{x}) = \sum_{\mathbf{y} \in X, \Pi_{c_i}\mathbf{y} = \Pi_{c_i}\mathbf{x}} p(\mathbf{y}) \tag{3.17}$$

$$P(\Pi_{b_i}\mathbf{x}|\Pi_{c_i}\mathbf{x}) = \frac{P(\Pi_{b_i}\mathbf{x}, \Pi_{c_i}\mathbf{x})}{P(\Pi_{c_i}\mathbf{x})} \tag{3.18}$$

Definition 10 *Given a set of sets $S = \{s_1, \ldots, s_l\}$, for $i = 1, \ldots, l$ we define:*
The histories

$$d_i = \cup_{j=1}^{i} s_j, \quad d_0 = \emptyset \tag{3.19}$$

the residuals

$$b_i = s_i \setminus d_{i-1} \tag{3.20}$$

and the separators

$$c_i = s_i \cap d_{i-1} \tag{3.21}$$

Theorem 4 *Let $P(\mathbf{x})$ be a Boltzmann distribution on $\mathbf{X}$ with*

$$P(\mathbf{x}) = \frac{u^{f(\mathbf{x})}}{\sum_y u^{f(y)}}$$

with arbitrary $u > 1$:
If

$$b_i \neq \emptyset \quad \forall i = 1, \ldots, l; \quad d_l = \tilde{X}, \tag{3.22}$$

$$\forall i \geq 2 \quad \exists j < i such\ that c_i \subseteq s_j \tag{3.23}$$

then

$$P(\Pi_{b_i}\mathbf{x}) = \prod_{i=1}^{l} P(\Pi_{b_i}\mathbf{x}|\Pi_{c_i}\mathbf{x}) \tag{3.24}$$

Therefore, when $f(\mathbf{x})$ is an ADF, the distribution can be computed in polynomial time using a decomposition of the join probability distribution using the *running intersection property* (Equation 3.23) [Lau96]. The factorization is supposed given, although its factorization process can be performed during the initialization of the FDA algorithm. The authors propose a simple factorization approach which assumes that the defining sets are sorted into a sequence $(s_1, \ldots, s_n)$. Then the sets b_i and c_i such that $b_i \neq \emptyset$ are computed according to the factorization theorem (Theorem 4). For the root set b_1, the sub function which is maximally nonlinear is chosen. The nonlinearity is measured as deviance from a linear square predictor). For faster convergence, the authors propose a local approximation $\tilde{P}(\mathbf{x})$of the true Boltzmann distribution $P(\mathbf{x})$. This approximation uses the same factorization that in the case of the true distribution, computing the conditional probabilities using the local fitness functions f_i:

$$\tilde{P}(\Pi_{b_i}\mathbf{x}|\Pi_{c_i}\mathbf{x}) = \frac{\tilde{P}(\Pi_{s_i}\mathbf{x})}{\tilde{P}(\Pi_{c_i}\mathbf{x})} = \frac{u^{f_i(\Pi_{s_i}\mathbf{x})}}{\sum_{\substack{\mathbf{y} \in \mathbf{X}_{s_i} \\ \Pi_{c_i} y = \Pi_{c_i}\mathbf{x}}} u^{f_i(\Pi_{s_i}\mathbf{y})}} \tag{3.25}$$

with $u \geq 1$. The larger u becomes, the *steeper* the distribution becomes. $u = 1$ yields a uniform distribution. The authors propose to chose u as:

$$\frac{1}{10} \leq \frac{\tilde{P}(\Pi_{b_i}\mathbf{x}|\Pi_{c_i}\mathbf{x})}{\tilde{P}(\Pi_{b_i}\mathbf{y}|\Pi_{c_i}\mathbf{y})} \leq 10 \quad i = 1, \ldots, l$$

by setting

$$\alpha = \max_i \left\{ \max_{\mathbf{x},\mathbf{y}} |f_i(\mathbf{x}) - f_i(\mathbf{y})| \right\}$$

$$u = 10^{\frac{1}{\alpha}}$$

The pseudo-code for the FDA algorithm is presented in Algorithm 16.

Algorithm 16 The *Factorized Distribution Algorithm*

```
Input: Evaluation function $f(\mathbf{x})$, initial factorization, and $r$ the percentage of individuals generated using local approximation.
  Set $l \leftarrow 0$
  Generate a random initial population $D_0$ with $(1-r)*M > 0$ individuals.
  Generate $r*M$ individuals using the local approximation (Equation 3.25).
  $\boxed{\textit{Label 1:}}$
  Select the promising points according their evaluation vale $f(\mathbf{x})$
  Compute the conditional probabilities $P^s(\Pi_{b_i}\mathbf{x}|\Pi_{c_i}\mathbf{x}, l)$ using the selected points.
  Generate a new population according to $P(\mathbf{x}, l+1) = \prod_{i=1}^{n} P^s(\Pi_{b_i}\mathbf{x}|\Pi_{c_i}\mathbf{x}, l)$.
  if The termination criteria is met then
    Goto Label 2
  end if
  Add the best point to the previous generated points (elitist).
  $l \leftarrow l+1$
  Goto Label 1
  $\boxed{\textit{Label 2:}}$
Output: Best individual found during the process.
```

EBNA

The *Estimation of Bayesian Networks Algorithm* was introduced by Etxeberria and Larrañaga in [EL99]. In contrast of previous algorithms, in this case, EBNA refers to a set of different algorithms based on the same underlying schemata, the construction of a probabilistic graphical model with no restriction in the number of parents that variables can have. We first introduce the underlying idea and finally three algorithms are described. For simplicity, references to EBNA refers to the common parts of the set of algorithms.

EBNA is based on the penalized maximum likelihood score. In this algorithm, given a population D with N samples, $D = \{x_1, \ldots, x_N\}$, a measure of the success of any structure S to describe the individuals in D is proposed. This measure is obtained by computing the maximum likelihood estimate $\hat{\theta}$ for the parameters θ and the associated maximized log-likelihood, $\log P(D|S, \hat{\theta})$. The main idea in EBNA is to search for the structure that maximizes $\log P(D|S, \theta)$ using an appropriate search strategy. The theoretical foundations of this intuitively appealing approach are based on the consistency and the asymptotic efficiency properties of the maximum likelihood estimates.

Let $X = (X_1, \ldots, X_n)$ be a set of random variables, and let x_i be a value of X_i, the i-th component of X. Then, a probabilistic graphical model for X is a graphical factorization of the joint generalized probability density function, $P(X = x)$ (or simply $P(x)$). The representation of this model is given by two components: a structure and a set of local generalized probability densities.

With regard to the structure of the model, the structure S for X is a *directed acyclic graph* (DAG) that describes a set of conditional (in)dependencies about the variables on X. $\mathbf{Pa}_i^S$ represents the set of parents (variables from which an arrow is coming out in S) of the variable X_i in the probabilistic graphical model whose structure is given by S. The structure S for $\mathbf{X}$ assumes that X_i and $\{X_1, \ldots, Xi-1\} \setminus \{\mathbf{Pa}_i^S\}$ are independent given $\mathbf{Pa}_i^S$, $i = 2, \ldots, n$. Therefore, the factorization can be written as follows:

$$P(\mathbf{x}) = P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | \mathbf{Pa}_i^S) \tag{3.26}$$

A representation of the models of the characteristics described above assumes that the local generalized probability densities depend on a finite set of parameters $\theta_S \in \Theta_S$, and as a result, the previous equation can be rewritten as follows:

$$P(\mathbf{x}|\theta_S) = \prod_{i=1}^{n} P(x_i | \mathbf{Pa}_i^S, \theta_i) \tag{3.27}$$

where $\theta_S = (\theta_1, \ldots, \theta_n)$.

With the previous definitions, the model can by represented by $M = (S, \theta_S)$. In the particular case of every variable $X_i \in \mathbf{X}$ being discrete, the probabilistic graphical model is called Bayesian network. If the variable X_i has r_i possible values, $\{x_i^1, \ldots, x_i^{r_i}\}$, the local distribution, $P(x_i | \mathbf{Pa}_i^{j,S}, \theta_i)$ is an unrestricted discrete distribution:

$$P(x_i | \mathbf{pa}_i^{j,S}, \theta_i) = \theta_{x_i^k | pa_i^j} \equiv \theta_{ijk} \tag{3.28}$$

where $pa_i^{1,S}, \ldots, pa_i^{q_i,S}$ denotes the values of $\mathbf{Pa}_i^S$, and $q_i = \prod_{X_g \in \mathbf{Pa}_i} r_g$ is the number of different possible instantiations of the parent variables of X_i. In other words, θ_{ijk} represents the conditional probability of variable X_i to take its K-th value x_i^k, knowing that the set of its parent variables take thir j-th combination values. It is assumed that $\theta_{ijk} > 0$.

Following the previous notation, the algorithm for different instances of the EBNA algorithm is presented in Algorithm 17. The join probability distribution in EBNA is represented by means of a Bayesian network, which is estimates each iteration from selected individuals of the population. The initialization of EBNA estimates an initial distribution model BN_0 where all the points in the search space have equal probability. This initial model is represented as a graph without edges, where the join probability distribution is factorized by means of the product of the n uniform marginal distributions.

The different instances of EBNA are obtained by varying the structural search method. More concisely, the authors propose two approaches to deal with the structure: The detection of conditional (in)dependencies using the *PC* algorithm [SGS00] and two *score+search* methods, the *Bayesian Information Criterion* (BIC) [Sch78] score and the $K2$ [CH92] + penalization score. The different instantiations for the EBNA algorithm are referred as: $EBNA_{PC}$, $EBNA_{K2+search}$, and $EBNA_{BIC}$ respectively. An extended information about these methods can be found in [Lar01].

Finally, the method used in order to generate new individuals from the model is the *probabilistic logic sampling* (PLS) [Hen88]. Following this method, the instantiations are done one variable at a time in a forward way, that is, a variable is not sampled until all its parents have already been so. This requires previously to order all the variables from parents to children. Once the values of the parent variables of a variable X_i have been assigned, the values for X_i will be simulated using the distribution $P(x_i|\mathbf{Pa}_i)$.

Algorithm 17 The *Estimation of Bayesian Networks Algorithm*, with the variants $EBNA_{PC}$, $EBNA_{K2+pen}$, and $EBNA_{BIC}$

```
Input: Evaluation function $f(\mathbf{x})$, the number of individuals $M$ of a population, and
  the number $S < M$ of individuals to be selected.
  $BN_0 \leftarrow (S_0, \theta^0)$ where $S_0$ is an arcless DAG, and $\theta^0$ is uniform over all the points.
  $P_0(\mathbf{x}) = \prod_{i=1}^{n} P(x_i) = \prod_{i=1}^{n} \frac{1}{r_i}$, where $r_i$ is the cardinality of $X_i$.
  Set $l \leftarrow 0$
  Generate an initial population $D_0$ with $M$ individuals, sampling from $P_0(\mathbf{x})$
  while Stopping criteria are met do
    Set $l \leftarrow l + 1$
    Select a subpopulation $D^{\star}_{l-1}$, selecting $N$ individuals from $D_{l-1}$ according to
    $f(\mathbf{x})$.
    Find the best structure $S^{\star}_l$ according to a criterion:

               $EBNA_{PC}$ :      conditional (in)dependence tests
               $EBNA_{K2+pen}$ :  penalized Bayesian score+search
               $EBNA_{BIC}$ :     penalized maximum likelihood+search

    Set $\theta^l$ as $\theta_{ijk}$ calculated using $D^{\star}_{l-1}$ as data set.
    $BN_l \leftarrow (S^s_l tar, \theta^l)$
    $D_l \leftarrow$ Sample $M$ individuals from $BN_l$ using PLS
  end while
Output: Best individual found during the process.
```

BOA

The *Bayesian Optimization Algorithm* (BOA) was introduced by Pelikan and Goldberg in [PGCP99], filling the gap between the fully informed FDA and totally uninformed black-box optimization methods. The combination of prior information and the set of promising solutions is used to estimate the distribution. Although prior information is not essential, the prior information about the structure of a problem as well as the information represented by the set of high quality solutions can be incorporated to the generation of new solutions.

As in many other EAPMs, the first population in the BOA is generated at random. From the current population the better individuals according an evaluation function are selected. A Bayesian network that describe the selected set of individuals is estimated using the *Bayesian Dirichlet equivalence* (BD_e) [HGC95] in order to measure the quality of the networks. This metric combines the prior knowledge about the problem and the statistical data from a given data set. The BD_e metric for a network B given a data set D of size N and the background information ξ denoted by $P(D, B|\xi)$ is defined as

$$P(D, B|\xi) = P(B|\xi) \prod_{i=1}^{n-1} \prod_{\mathbf{Pa}_i} \frac{m'(pa_i)!}{(m'(pa_i) + m(pa_i))!} \prod_{x_i} \frac{(m'(x_i, pa_i) + m(x_i, pa_i))!}{m'(x_i, pa_i)!} \tag{3.29}$$

where $P(B|\xi)$ is the prior probability of the network B, the product over pa_i runs over all instances of the parents of X_i and the product over x_i runs over all instances of X_i. By $m(pa_i)$, the number of instances in D with variables $\mathbf{Pa}_i$ (the parents of X_i) instantiated to pa_i is denoted. When the set $\mathbf{Pa}_i$ is empty, there is one instance of $\mathbf{Pa}_i$ and the number of instances with $\mathbf{Pa}_i$ instantiated to this instance is set to N. By $m(x_i, pa_i)$ we denote the number of instances in D that have both X_i set to x_i as well as $\mathbf{Pa}_i$ set to pa_i.

By numbers $m'(x_i, pa_i)$ and $P(B|\xi)$ prior information about the problem is incorporated into the metric. The $m'(x_i, pa_i)$ stands for prior information about the number of instances that have X_i set to x_i and the set of variables $\mathbf{Pa}_i$ is instantiated to pa_i. The prior probability $P(B|\xi)$ of the network reflects how the measured network resembles the prior network. By using a prior network, the prior information about the structure of a problem is incorporated into the metric. The prior network can be set to an empty network, when there is no such information. If no prior information is available, $P(B|\xi)$ can be set to one for all networks, therefore, all networks are treated equally.

The numbers $m'(x_i, pa_i)$ can be set in various ways. They can be set according to the prior information the user has about the problem. When there is no prior information, uninformative assignments can be used. In the so-called $K2$ metric [CH92], for instance, the $m'(x_i, pa_i)$ coefficients are all simply set to one. This assignment corresponds to having no prior information about the problem.

In order to find the network which maximize the metric value, any search algorithm can be used. A new population is generated using the joint distribution encoded by the constructed network and some individuals from the old population are replaced with the new ones. The Pseudo-code of the BOA is shown in Algorithm 18.

Algorithm 18 The *Bayesian Optimization Algorithm*

```
Input: Evaluation function $f(\mathbf{x})$, the number of individuals $M$ of a population, and
  the number $S < M$ of individuals to be selected.
  Set $t \leftarrow 0$
  Generate a random population $D_0$ with $M$ individuals
  while Stopping criteria are not met do
    Select a set of $S$ promising individuals $D_0^\star$ according to $f\mathbf{x}$.
    Construct the network $B$ using a chosen metric and constraints.
    Generate a set of new individuals $D'_t$ according to the joint distribution encoded
    by $B$
    Create a new population $D_{t+1}$ by replacing some individuals from $D_t$ with $D'_t$.
    Set $t \leftarrow t + 1$
  end while
Output: Best individual found during the process.
```

3.3.5 EBCOAs

Evolutionary Bayesian Classifier-based Optimization Algorithms(EBCOAs) were introduced by Miquélez et al. in [MBL04]. This new paradigm in the evolutionary computation is an improvement of the EAPM motivated for the need to avoid them to fall into local optima in very complex optimization problems. The main difference between the different evolutionary strategies presented in previous sections, is the way of improving the population of individuals in order to obtain better solutions to a concrete optimization problem: In Genetic algorithms the evolution is based on using crossover and mutation operators, without expressing explicitly the characteristics of the selected individuals within a population. EAPMs take into account these explicit characteristics by considering the interdependencies between the different variables that defines an individual, learning a probabilistic model to represent them.

EBCOAs innovative contribution is twofold: firstly, it evolves a generation of individuals by constructing Bayesian classifier models that take into account deeper differences rather than simply a subset of individuals of the previous population. Secondly, it also takes into account the differences between individuals in the population that make them more or less fit regarding their fitness values, and it applies this knowledge to create a new population by enhancing the characteristics of the better individuals and tries to avoid the less fitted ones [MBL04]. Summarizing, this new approach propose the use of classification techniques in the form of Bayesian networks applied to optimization problems in order to improve the generation of individuals in every iteration.

Let $\mathbf{X} = (X_1, \ldots, X_n)$ be an $n-$dimensional random variable. Then $\mathbf{x} = (x_1, \ldots, x_n)$ represents one of its possible instantiations and therefore one of the possible individuals. The probability of $\mathbf{X}$ is denoted by $P(\mathbf{X} = \mathbf{x})$, or simply $P(\mathbf{x})$. The conditional probability of the variable X_i given the value x_j of the variable X_j is denoted as $P(X_i = x_i | X_j = x_j)$, or simply as $P(x_i|x_j)$. Let D_t be the $t-$th population of the M individuals that has to evolve into the $(t+1)$-th one. In EBCOAs, before proceeding to the learning, the population D_t is divided into $|K|$ different classes following a supervised classification approach, and we define a variable $K \in \{1, 2, \ldots, |K|\}$. The result of divide D_t into the $|K|$ groups is denoted as D_t^K, and for each individual in the population a value k is assigned to the variable K in order to represent the class to which each individual has been assigned. Since all the classes are not usually used for the learning, prior to training the Bayesian classifier we choose $|C| < |K|$ classes and the rest are simply ignored for learning purposes. The Pseudo-code for EBCOA is shown in Algorithm 19.

The supervised classification problem consists of assigning a vector $x = (x_1, \ldots, x_n)$ to one of the $|C|$ classes of variable C. The true class is denoted by c and it takes values from the set $\{1, 2, \ldots, |C|\}$. The classifier can be seen as a function $\gamma : (x_1, \ldots, x_n) \mapsto \{1, \ldots, |C|\}$ that assigns labels to observations. Miquélez et al. [MBL04] propose four different Bayesian classifiers:

Naïve Bayes: This paradigm combines the Bayes theorem with the assumption that all the variables are independent given the class. This Bayesian network has always the same structure: all variables $X_1 \ldots, X_n$ are considered to be condi-

Algorithm 19 Pseudo-code for *Evolutionary Bayesian Classifier-based Optimization Algorithms.*

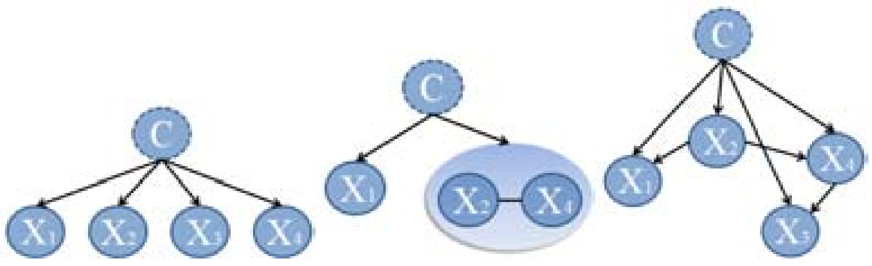
```
Input: Evaluation function $f(\mathbf{x})$, the number of individuals $M$ of a population, and
  the number $|K| < M$ of classes into which the individuals will be split.
  Set $t \leftarrow 0$
  Generate a random population $D_0$ with $M$ individuals.
  while Stopping criteria are not met do
    $D_t^K \leftarrow$ Divide the $M$ individuals in $|K|$ different classed from $D_t$ according to a
    criterion
    $D_t^C \leftarrow$ Select the $|C| \leq |K|$ classed of $D_r^k$ that will be used for building the
    Bayesian classifier, usually taking into account at least the best and worst classes.
    The individuals of the classes not included in $D_t^C \subset D_t^K$ are ignored.
    $P_t(c|\mathbf{x}) \propto P_t(\mathbf{x}|c) \leftarrow$ Estimate the probability distribution of an individual in $D_t^C$
    of being part of any of the different possible $|C|$ classes.
    $D_{t+1} \leftarrow$ Sample $M$ new individuals from $P_t(c|\mathbf{x})$
    Set $t \leftarrow t + 1$
  end while
Output: Best individual found during the process.
```

tionally independent given the class value C(see Fig. 3.12a).

Selective naïve Bayes: The main difference between selective naïve Bayes and naïve Bayes is that in selective naïve Bayes not all variables have to be present in the final model. Missing variables are not considered in the classification process, therefore, their node do not appear in the graphical model.

Semi-naïve Bayes: The semi-naïve Bayes classifier provides more complexity than previous ones since it is able to take into account dependencies between groups of variables. This paradigm represents the variables found to be related as a fused node in the conditional Gaussian network, that is the semi-naïve Bayesian classifier proposed to group some variables in a single node of the structure (see Fig. 3.12b). When grouping variables all the inter-dependencies between them are taken into account implicitly in the Bayesian classifier. Two greedy algorithms are presented in order to build the model, the first of them in a forward direction called FSSJ (Forward Sequential Selection and Joining), and the second in the backward direction named BSEJ (Backward Sequential Elimination and Joining).

Tree augmented naïve Bayes: The last approach is based on tree augmented naive Bayes, a Bayesian network classifier in which the dependencies between variables other than C are also taken into account. These models represent the relationships between the variables $X_1, \ldots, X_n$ conditional on the class variable C by using a tree structure (see Fig. 3.12c). The tree augmented naive Bayes structure is built in a two-phase procedure: Firstly, the dependencies between the different variables $X_1, \ldots, X_n$ are learned using a score based on information theory. In the second phase, the structure is augmented into the naïve Bayes paradigm.



(a) Naïve Bayes (b) Semi-naïve Bayes (c) Tree Augmented Naïve Bayes

Figure 3.12: Different structures of Bayesian classifiers considered for different classification model building algorithms in a problem with four variables $X_1, \ldots, X_4$ and the class variable C.

3.4 Standard problems

The evolutionary computation community has defined some standard problems in order to study the performance of new algorithms in certain conditions. In this section we introduce some of the most representative problems. Since our work is fundamentally based on discrete algorithms, we only introduce the problems based on discrete variables.

3.4.1 One Max

This is one of the most used and simple linear problem. It can be defined as:

$$F_{OneMax}(\mathbf{x}) = \sum_{i=1}^{n} x_i \tag{3.30}$$

where $x_i \in \{0, 1\}$. The global optimum is located at the point $(1, \ldots, 1)$.

3.4.2 Plateau

This problem was proposed by Mühlenbein and Schlierkamp-Voosen in [MSV93]. The individuals of this function consists of a $n-$dimensional vector, such that $n = m \times k$, where the genes are divided into groups of k bits. The problem can be defined as:

$$F_{Plateau}(\mathbf{x}) = \sum_{i=1}^{m} g(\mathbf{s}_i) \tag{3.31}$$

where $\mathbf{s}_i = (x_{ki-(k-1)}, x_{ki-(k-2)}, \ldots, x_{ki})$, and g is an auxiliary function defined as:

$$g(x_1, \ldots, x_k) = \begin{cases} 1 & \text{if} \quad x_1 = x_2 = \ldots = x_k = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.32}$$

As in the previous function, the goal is to maximize the function $F_{Plateau}$ and the global optimum is located at the point $(1, \ldots, 1)$.

3.4.3 Checkerboard

This problem was proposed by Baluja and Davies in [BD97b]. In this problem, a $s \times s$ grid is given. Each point of the grid can take a value 0 or 1. The goal is to create a checkerboard pattern of 0s and 1s. Each point with a value of 1 should be surrounded in all directions by points with a value 0, and viceversa. The evaluation counts the number of correct surrounding bits. The corners are not included in the evaluation. The maximum value is $4 \times (s-2)^2$, and the problem dimension is $n = s^2$. If the grid is considered as a matrix $\mathbf{x} = [x_{i,j}]_{i,j=1,\ldots,s}$ and defines $\delta(a, b)$ as the Kronecker's delta function,

$$\delta(i,j) = \begin{cases} 0 & \text{for } i \neq j \\ 1 & \text{for } i = j \end{cases} \tag{3.33}$$

the checkerboard function is written as:

$$F_{CheckerBoard}(\mathbf{x}) = 4(s-2)^2 - \\ \sum_{i=2}^{s-1} \sum_{j=2}^{s-1} \{\delta(x_{i,j}, x_{i-1,j}) + \delta(x_{i,j}, x_{i+1,j}) + \delta(x_{i,j}, x_{i,j-1}) + \delta(x_{i,j}, x_{i,j+1})\} \tag{3.34}$$

3.4.4 Equal Products

As in the case of *Checkerboard* problem, this problem was proposed by Baluja and Davies in [BD97b]. Given a set of n random real numbers $\{a_1, \ldots, a_n\}$ from an interval $[0, k]$, a subset of them is selected. The aim of the problem is to minimize the difference between the products of the selected and unselected numbers. The evaluation function for this problem can be written as:

$$F_{EqualProducts}(\mathbf{x}) = \left| \prod_{i=1}^{n} h(x_i, a_i) - h(1 - x_i, a_i) \right| \tag{3.35}$$

where function h is defined as:

$$h(x,a) = \begin{cases} 1 & \text{if } x = 0 \\ a & \text{if } x = 1 \end{cases} \tag{3.36}$$

The optimum value is unknown because the set of real numbers is random, however, better results are found near the zero.

3.4.5 Six Peaks

This problem was also defined by Baluja and Davies in [BD97b]. In this case, the authors propose a hard optimization function which is defined as:

$$F_{SixPeacks}(\mathbf{x}) = \max\{tail(0, \mathbf{x}), head(1, \mathbf{x}), tail(1, \mathbf{x}), head(0, \mathbf{x})\} + \mathcal{R}(\mathbf{x}, t) \tag{3.37}$$

where

$$\begin{aligned} \mathit{tail}(b, \mathbf{x}) &= \text{ number of trailing } b\text{'s in } \mathbf{x} \\ \mathit{head}(b, \mathbf{x}) &= \text{ number of leading } b\text{'s in } \mathbf{x} \end{aligned}$$

$$\mathcal{R}(\mathbf{x}, t) = \begin{cases} n & \text{if } (\mathit{tail}(0,\mathbf{x}) > t \text{ and } \mathit{head}(1,\mathbf{x}) > t) \text{ or} \\ & (\mathit{tail}(1,\mathbf{x}) > t \text{ and } \mathit{head}(0,\mathbf{x}) > t) \\ 0 & \text{otherwise} \end{cases}$$

The goal is to maximize the function, obtaining one of their four global optima, located at the points:

$$(\overbrace{0,0,\ldots,0}^{t+1},1,1,\ldots,1) \quad (\overbrace{1,1,\ldots,1}^{t+1},0,0,\ldots,0)$$
$$(0,0,\ldots,0,\overbrace{1,1,\ldots,1}^{t+1}) \quad (1,1,\ldots,1,\overbrace{0,0,\ldots,0}^{t+1})$$

These points are difficult to obtain because they are isolated and in addition, there are two easily reachable local optima at $(0, 0, \ldots, 0)$ and $(1, 1, \ldots, 1)$. The value of t can be set in order to modify the difficulty of the problem, but it is usually set to $\frac{n}{2} - 1$.

3.4.6 HIFF

The *Hierarchical-if-and-only-if* (HIFF) was proposed by Watson and Pollack in [WP99]. The individuals of this problem are defined to be $n-$dimensional binary vectors (with $n = 2^j$), which bits are hierarchically grouped. The fitness value for the HIFF is defined as a recursive function which interprets the individuals as a binary tree and recursively decomposes the individual into left and right halves. In this manner, a string is evaluated by summing the fitness contributions of all sub-blocks at all levels:

$$F_{HIFF} = \begin{cases} 1 & \text{if } |B| = 1 \\ |B| + F_{HIFF}(B_L) + F_{HIFF}(B_R) & \text{if } |B| > 1 \text{ and } (\forall i | b_i = 0 \text{ or } \forall i | b_i = 1) \\ F_{HIFF}(B_L) + F_{HIFF}(B_R) & \text{otherwise} \end{cases} \tag{3.38}$$

where B is a block of bits, $\{b_1, \ldots, b_k\}$, $|B|$ is the size of the block ($|B| = k$), b_i is the $i-$th element of B, and $B_L = \{b_1, \ldots, b_{\frac{k}{2}}\}$ and $B_R = \{b_{\frac{k}{2}+1}, \ldots, b_k\}$ are the left and right halves of B.

Local optima in HIFF occur when incompatible building-blocks are brought together. For example, consider $(1, 1, 1, 1, 0, 0, 0, 0)$ viewed as two blocks from the previous level (i.e. size 4) both blocks are good -each contains one of the two global optima - but when these incompatible blocks are put together they create a sub-optimal string that is maximally distant from the next best strings i.e. $(1, 1, 1, 1, 1, 1, 1, 1)$ and $(0, 0, 0, 0, 0, 0, 0, 0)$.

3.4.7 IsoPeak

The IsoPeak problem was proposed by Mahnig and Mühlenbein in [MM01]. The individuals for this function consists of a $n-$dimensional vector, such that $n = 2 \times m$

(the genes are divided into groups of two). Using the auxiliary functions $Iso1$ and $Iso2$ defined in Table 3.1, the $IsoPeak$ function is:

Table 3.1
Auxiliary functions in the IsoPeak problem

x	00	01	10	11
$Iso1$	m	0	0	$m-1$
$Iso2$	0	0	0	m

$$F_{IsoPeak}(\mathbf{x}) = Iso2(x_1, x_2) + \sum_{i=2}^{m} Iso1(x_{2i-1}, x_{2i}) \tag{3.39}$$

The goal is to maximize the function $F_{IsoPeak}$ and the global optimum is located at the point $(1, 1, 0, 0, \ldots, 0)$.

3.4.8 Isotorus

This function can be found in the work of Miquelez et al. [MBL04]. The individuals for this function consists of a binary $n-$dimensional vector. Given $n = m^2$, and

$$IsoT1(u) = \begin{cases} m & \text{if } u = 0 \\ m-1 & \text{if } u = 5 \\ 0 & \text{otherwise} \end{cases}$$

$$IsoT2(u) = \begin{cases} m^2 & \text{if } u = 5 \\ 0 & \text{otherwise} \end{cases}$$

this function can be formulated as:

$$\begin{aligned} F_{IsoTorus}(\mathbf{x}) = \quad & IsoT1(x_{1-m+n} + x_m + x_1 + x_2 + x_{1+m}) + \\ & \textstyle\sum_{i=2}^{n} IsoT2(x_{up} + x_{left} + x_i + x_{right} + x_{down}) \end{aligned} \tag{3.40}$$

where x_{up}, x_{left}, x_{right}, and x_{down} are the appropriate neighbors of x_i.

3.5 Results and Conclusions

After a wide introduction to the most relevant algorithms in the state of the art of evolutionary computation, in this section different evolutionary algorithms are compared. Since in the literature authors use different subsets of these problems or either specific problems to compare their algorithms with standard ones, some evolutionary algorithms which code is available online are tested with the standard problems described in Section 3.4.

In order to us the same parameters in all the tests, we select a problem dimension and population size compatible with the imposed requirements of all algorithms and problems. The population size has been fixed at 120 individuals, and the dimension to 64 bits. The algorithms and specific parameters are detailed in the following:

Genetic Algorithms: We use the Matlab Optimization Toolbox for the Genetic Algorithms, using a Gaussian based mutation probability (the Gaussian is centered at zero with a variance of the half of the variable range, decreasing the variance along the generations), and scattered cross-over strategy with a cross-over fraction of 0.8.

PBIL: The PBIL algorithm has been programmed in Matlab, using a learning rate of 0.1.

UMDA: We use the code published by Roberto Santana [San].

EcGA: In order to apply the EcGA, we use the code of Sastry [SOP07].

Al those standard evolutionary algorithms are used to optimize each problem 10 times, and the mean value is shown in Table 3.2.

Table 3.2
Comparative of different evolutionary algorithms over standard problems. For each problem the optimal value and the type of optimization (maximization or minimization) are detailed.

	MeanValue			
Problem *(Value, Type)*	**GA**	**PBIL**	**UMDA**	**EcGA**
OneMax (64,max)	60.4(3)	64(1)	61.5(2)	42.8(4)
Plateau (16,max)	15.80(4)	16(2)	16(2)	16(2)
Checkerboard (144,max)	144(2.5)	144(2.5)	144(2.5)	144(2.5)
Six Peaks (123,max)	33.4(2)	70.2(1)	20.9(3)	16.3(4)
HIFF (448,max)	218.6(3)	241.2(1)	204.6(4)	235(2)
IsoPeak (1024,max)	1024(2.5)	1024(2.5)	1024(2.5)	1024(2.5)
IsoTorus (505,max)	298.8(2)	440.8(1)	271.4.6(3)	119.7(4)
Mean Rank	2.7143	1.5714	2.7143	3

Although it is difficult to obtain the code to test some of the state-of-the-art methods, in the literature we can found good performance studies where different methods are compared. An interesting example is found in [MBL04], where authors compare the performance of different EBCOAs (see Section 3.3.5) with other standard Probability Darwin Machines. The results for these experiments are sumarized in Table 3.3, using the same problem dimension and runs that in the previous experiments.

In order to analyze the results in Tables 3.2 and 3.3, we use the statistical analysis developed by Demšar in [Dem06] (see Appendix D) for each table.

Let r_i^j be the rank of the j-th of k algorithms on the i-th of N data sets. The Friedman test compares the average ranks of algorithms, $R_j = \frac{1}{N}\sum_i r_i^j$. Under the null-hypothesis, which states that all the feature sets are equivalent, and so their average ranks R_j are equal, the Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{3.41}$$

Table 3.3
Mean value after 10 executions with each algorithm and objective function [MBL04]. Note that the authors use an alternative definition for the IsoPeak function, where the optimum value differs, but the complexity and best individual are maintained.

	Problem (Value,Type)			
	HIFF	IsoPeak	IsoTorus	**Mean**
Algorithm	(448,max)	(3907,max)	(505,max)	**Rank**
$\text{EBCOA}_{\text{nBayes}}$	290(9)	3906(5)	505(1)	5.1667
$\text{EBCOA}_{\text{selectivenBayes}}$	355.2(6)	3906(5)	472(8)	6.3333
$\text{EBCOA}_{\text{seminnB-FSSJ}}$	290.2(8)	3859.8(8)	471.6(9)	8.3333
$\text{EBCOA}_{\text{seminnB-BSSJ}}$	184.5(11)	3803.8(9)	474.3(7)	9
$\text{EBCOA}_{\text{TANB}}$	448(2)	3907(1)	505(1)	1.5
UMDA	295.6(7)	3905.5(7)	400.3(11)	8.3333
MIMIC	283.2(10)	3906(5)	422.3(10)	8.3333
EBNA	448(2)	3906.3(2)	485.2(5)	3
cGA	395.2(4)	3628.1(11)	477.2(6)	7
eGA	388.8(5)	3793.7(10)	488.5(3.5)	6.1667
ssGA	448(2)	3906.1(3)	488.5(3.5)	2.8333

is distributed according to χ_F^2 with $k-1$ degrees of freedom when N and k are big enough. For a small number of algorithms and data sets, exact critical values have been computed. Following the considerations of Iman and Davenport [ID80], we compute:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2} = \frac{4\times\chi_F^2}{5-\chi_F^2} \qquad (3.42)$$

which is distributed according to the F-distribution with $k-1=3$ and $(k-1)(N-1) = 3\times 6 = 12$ degrees of freedom. For data in Table 3.2, we obtain $\chi_F^2 = 5.1$ and $F_F = 1.9$. The critical value of $F(3,12)$ for $\alpha = 0.05$ is 3.16, which is larger than F_F, so we cannot reject the null-hypothesis at 95%. The critical value of $F(3,12)$ for $\alpha = 0.1$ is 2.42, even larger than F_F, therefore, null-hypothesis cannot be rejected at 90%. Therefore, all methods are statistically equivalents.

In the case of data in Table 3.3, we obtain $\chi_F^2 = 18.12$ and $F_F = 3.05$. The critical value of $F(10,20)$ for $\alpha = 0.05$ is 2.34, which is smaller than F_F, so we reject the null-hypothesis at 95%. Once the null-hypothesis has been rejected, we know that algorithms are not statistically equivalent, therefore, we can proceed with a post-hoc test. In our case, as no algorithm is singled out for comparisons, we use the Nemenyi test for pairwise comparisons. The performance of the two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference

$$CD = q_\alpha\sqrt{\frac{k(k+1)}{6N}} = 8.57 \qquad (3.43)$$

where critical values q_α are based on the Studentized range statistic divided by $\sqrt{2}$. Using averaged ranks in table 3.3 we calculate all the pair-wise differences. Since none of those differences is larger than the critical difference, we can conclude that although methods are not equivalent, the post-hoc test is not powerful enough to state differences between methods. However, looking at results in Tables 3.2 and 3.3, we can see that EBCOAs use to perform better than most of the others.

The fact that standard problems are prepared to contain some type of issues for the optimization methods, can explain the absence of a clear predominant method. Each method is designed in order to be robust in front of a certain issue, but it does not exist a perfect method. Because of that, in the literature we can find hybrid methods, which try to combine the benefits of different methods in one. For instance, we can create a coarse-to-fine search method using evolutionary algorithms for a coarse search over the whole solutions space and near better points apply a fine search using exhaustive methods or gradient based methods.

Finally, it is important to note that most standard problems are defined for binary problems, what is not a real limitation because any numerical problem can be represented using binary encoding. However, when a value is represented using a set of bits, a new dependency level is added to the problem, where apart of the inherent dependencies, bits representing the same value have an additional dependence. It can provoke that methods which have a good performance in a binary problem do not perform as well in numerical optimization. Next section describes the application of those methods to the object detection problem.

Chapter 4

Evolutionary object detection

Once the object detection framework and evolutionary computation have been described, this section explains how we merge both methodologies in order to cope with some of the limitations of the classical object detection approach introduced in Section 2.1.2. We first motivate the necessity of mixing those methodologies, highlighting the limitations of the AdaBoost algorithm, and in further sections, we build the necessary framework in order to deal with an evolutionary version of the AdaBoost algorithm. We first develop a system using Haar-like features and Genetic Algorithms, establishing a reference framework that will be progressively improved, first by exploring new features and finally new evolutionary methods based on Probabilistic Darwin Machines.

4.1 Motivation

Rectangle features are somewhat primitive when compared with alternatives such as steerable filters [FA91]. Steerable filters, and their relatives, are excellent for the detailed analysis of boundaries, image compression, and texture analysis. In contrast, rectangle features, while sensitive to the presence of edges, bars and other simple image structure, are quite coarse. Unlike the steerable filters, the orientation of the rectangle features is restricted to a few orientations.

In spite of their simplicity, rectangular features provide a rich image representation which supports effective learning. The extreme computational efficiency of rectangular feature provides ample compensation for their limited flexibility. Rectangle features have the property that a single feature can be evaluated at any scale and location in a few operations.

It is important to note that Haar features constitute an overcomplete dictionary of the image and that there are more than 2^{18} different features for a small image window of 576 pixels (24×24 pixels). This fact imposes a high computational cost on the learning step of the Adaboost algorithm, which involves several rounds of exhaustive searches. From a practical point of view, the development of a high performance object detector represents, when using conventional hardware, a learning time of the order of several hundred hours.

The work of Viola & Jones [VJ01] was extended by Lienhart and Maydt [LM02], who showed that the use of a larger feature set may improve the convergence and the performance of the final classifier. The extension of the feature set was done by adding rotated versions of original Haar-like features, and thus adding a factor to the exponential relation between the size of the feature set and the training time.

In order to make feasible the use of extended feature sets, we need to redefine the classical approach in order to avoid exhaustive search over the feature space. Our proposal is to use evolutionary computation techniques to deal with the object detection problem.

4.2 Antecedents

Darwin Machines in the field of image processing, especially automatic learning of features for object detection, is a field of research which receives growing interest. Howard et al. [HRB99] apply Genetic Programming (GP) to build a classifier that detects ships in satellite images. Krawiec [KB07] extends standard GP by a local search operation for visual learning. Lin et al. [LB03] propose a co-evolutionary GP to learn composite features based on primitive features that are designed by human experts. Bala et al. [BJH+96] combine a Genetic Algorithm (GA) with decision tree learning: The GA selects a good subset of features from a fixed set and a decision tree is learned to build the detector structure. Guarda et al. [GGL98] combine a GA to select different convolution masks (features) with GP to evolve the final detector based logical combinations of pixel convolutions in subwindows.

The combination of Genetic Darwin Machines and AdaBoost has been exploded by different authors in the last years. Sedai & Rhee [SR07] proposed the use of Genetic Algorithms in order to create subsets of features which adapts better to certain classification tasks and combine that specific subsets using AdaBoost. Treptow et al. [TZ04] uses a Genetic Algorithm in order to evolve an extension of the Haar-like features, using AdaBoost to combine the evolved features. Despite in this latter work only Genetic Algorithms are proposed, the underlying idea is closely related to the work developed in this thesis. The rest of this chapter describes how the process of learning a detector can be approached as a function optimization problem. In addition, implementations using Genetic and Probabilistic Darwin Machines are developed.

4.3 From object detection to function optimization

Given a training set $\langle(\mathbf{x}_1, y_1), ..., (\mathbf{x}_M, y_M)\rangle$, where $y_i \in \{-1, +1\}$ is the target value for sample $\mathbf{x}_i$, the goal of an object detection learning algorithm is to deal with the inference of a strong classifier $H(\mathbf{x}_i) = y_i$. In the boosting framework, we define a distribution $W = \{w_1, ..., w_M\}$ over the training set, where each w_i is the weight associated to the sample $\mathbf{x}_i$, and $H(\mathbf{x})$ corresponds to an additive model $H(\mathbf{x}) = \sum_t \alpha_t h_t(\mathbf{x})$ where the final decision is a combination of the decisions of several *weak classifiers* $h(\mathbf{x}) \in \{-1, +1\}$. In contrast to the strong classifier $H(\mathbf{x})$ where we expect a good performance for any sample $\mathbf{x}_i$ in the training set, in the case of weak classifier we only expect it is better than a random decision.

Given $\mathcal{H}$ the set of all possible weak classifiers, $h^{\mathbf{s}} \in \mathcal{H}$ a certain weak classifier defined by parameters $\mathbf{s}$, W the weights distribution of the Adaboost and $\epsilon(h^{\mathbf{s}}) = Pr_{i \sim W}[h^{\mathbf{s}}(\mathbf{x}_i) \neq y_i]$ the error function, the regression step consists on finding $\mathbf{s}^*$ that $\epsilon(h^{\mathbf{s}^*}) \leq \epsilon(h^{\mathbf{s}}) | \forall h^{\mathbf{s}^*}, h^{\mathbf{s}} \in \mathcal{H}$, where the complexity of finding $\mathbf{s}^*$ depends on the size of $\mathcal{H}$. In Algorithm 20, the evolutionary version for the Discrete Adaboost is shown. Note that the only difference with the classical Discrete AdaBoost relies on how the weak hypothesis h_t is obtained. The same approach can be used in all the variants of AdaBoost.

Algorithm 20 Evolutionary Discrete Adaboost

```
Input: A training set $X$ of $N$ pairs $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ is the $i^{th}$ image and $y_i \in \{0,1\}$
  is the category of the object present in $\mathbf{x}_i$, an evolutionary weak learning algorithm
  (EvolWeakLearner) and the maximum number of iterations $M$.

  Initialize the weight vector: $w_i^1 = \frac{1}{N}$ for $i = 1, ..., N$.
  for $t = 1, .., M$ do
    Set
        $$p^t = \frac{w^t}{\sum_{i=1}^{N} w_i^t}$$
    Call the evolutionary method EvolWeakLearner, providing it with the distri-
    bution $\mathbf{p^t}$. Get back a hypothesis $h_t : X \rightarrow [0,1]$. which minimize:
        $$\epsilon_t = Pr_{i \sim W_t}[h_t(x_i) \neq y_i]$$
    Get weak hypothesis $h_t(x) \mapsto \{-1, +1\}$ with error $\epsilon_t$
    Update:
        $$W_{t+1}(i) \leftarrow W_t(i) \times \exp(-y_i \times h_t(x_i))$$
    Normalize $W_{t+1}$ so
        $$\sum_{i=1}^{M} W_{t+1}(i) = 1$$
  end for

Output: the final hypothesis:
        $$H(x) = sign\left(\sum_{t=1}^{T} h_t(x)\right)$$
```

Once the evolutionary approach is defined, in order to learn an object detection we need to define what is $\mathbf{s}$. In general terms, $\mathbf{s}$ is the set of all the parameters that defines a *weak hypothesis*, and thus, it is closely related to the features we use to describe the objects and how decisions are made. Therefore, we can divide $\mathbf{s} = \{s_1, ..., s_D\}$ into two different subsets $\mathbf{s_M} = \{s_1, ..., s_i\}$ and $\mathbf{s_F} = \{s_{i+1}, ..., s_D\}$, containing the parameters of the decision method and the parameters of the features respectively.

Features parametrization

The first step on object detection is to choose a description method for the images. No restrictions on the type of features is assumed, therefore, any of the features introduced in Section 2.3, or that else descriptors we can imagine can be used to describe objects. In general, there are some feature-specific parameters that must be defined in order to use each descriptor (i.e. regions of a Haar-like feature or Dissociated Dipole, number of bins in the SIFT and SURF descriptors, etc...). These parameters can be discrete or continuous, predefined or learned, etc... All those parameters are included into $\mathbf{s_F}$. Given an instance for each parameters, we must be able to represent an object, either using a single value (i.e. Haar-like) or a vector (i.e. SIFT).

Classifier parametrization

Once the object is described, a decision must be performed using the descriptor associated to the object. Although there are different approaches to generate hypothesis from descriptors, in general we can consider threshold based decisions when objects are described by a single value, and some type of distance metric with vectors. On the first case, the threshold value and the polarity value must be included as parameters. In the second case, parameters to define the metric or reference points can be added on the parameters. Moreover, once the distance is evaluated, decision should be generated using a threshold like approach, and thus, the threshold and polarity parameters will also be included as classifier parameters $\mathbf{s_M}$.

4.4 Evolutionary object detection approach

At this point, object detection approach based on a boosting strategy has been redefined as the problem of finding the parameters $\mathbf{s}$ that minimize the weighted error function $\epsilon(h^{\mathbf{s}})$, that is, an optimization problem. The classical approaches perform an exhaustive search over parameters space in order to find out the best values. Although this can be done when the number of values that parameters can take is reduced, it becomes unfeasible in large search spaces.

In the literature we find many different approaches to deal with optimization problems with large search spaces, most of them based on gradient descent, as line search methods, normalized steepest methods or the Newton steps method. In those methods, the goal function is required to be differentiable, and uses the gradient direction to move from a certain solution to a better one. In general, that restriction can not be guaranteed for the error function, where small changes on the parameters can produce large discontinuities on the error function. On this scenario, the most common optimization methodologies are based on evolutionary computation, and in general, the first choice are Genetic Algorithms.

Once a general formulation and its considerations are provided, the first step is to verify that using evolutionary strategies we can obtain the same results that in the case of exhaustive methods, therefore, we need to define a framework where both approaches can be applied and compare their learning capabilities. In the following, a parametrization for Haar-like features using decision stumps is defined, analogously to the Viola's & Jones approach introduced in Section 2.7.

4.4.1 Classifier implementation

In order to predict the classifier value of a certain object, a decision stump will be used in all the experiments. As we saw in Section 2.4, is a linear classifier that uses an hyperplane to classify points in a binary problem. The parameters related to a decision stump is the hyperplane parameters which codifies how the space is divided (see Fig. 4.1) and a polarity value which decides which class is at each side of the hyperplane (see Fig. 4.2).

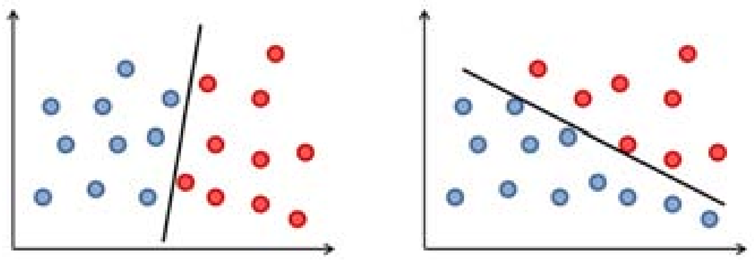


Figure 4.1: Two different instances of the hyperplane parameters. It exists an infinity of possible instances for the hyperplane parameters.

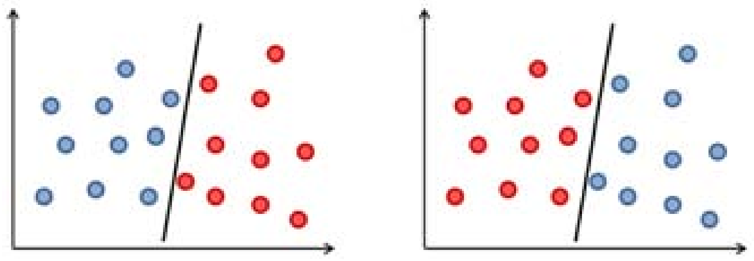


Figure 4.2: The polarity value codifies which side of the hyperplane corresponds to each class. There are only two possible values for this parameter.

Given an object descriptor $\mathbf{d} \in \mathbb{R}^N$, classifier parameters can be defined as $\mathbf{s_M} = \{s_p \in \{-1, +1\}, \mathbf{s}_h \in \mathbb{R}^N\}$, where s_p is the polarity value and $\mathbf{s_h}$ the parameters of the hyperplane.

4.4.2 Features implementation

Object description will be performed using Haar-like features used in the work of Viola & Jones (see Section 2.3.1). Given the regions configuration, a feature is determined only with one of their regions. In Fig. 4.3 the different considered configurations are presented.

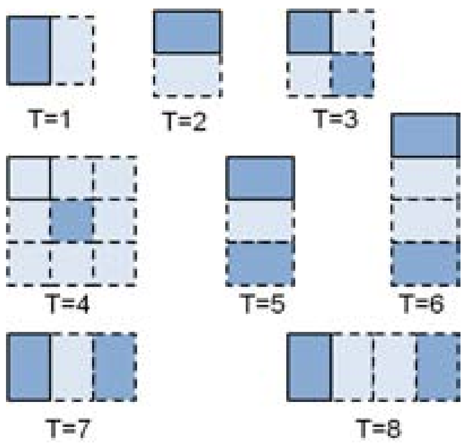


Figure 4.3: Different Haar-like region configurations. Dashed lines corresponds to regions inferred from the given region, which are represented with a continuous line. Darker regions correspond to inhibitory (negative) regions while lighter ones are the excitatory (positive) regions.

Therefore, codifying a Haar-like feature is equivalent to codify just one rectangle and the configuration. The codification of a rectangle can be easily performed by the codification of their upper-left corner and its vertical and horizontal sizes (see Fig. 4.4).

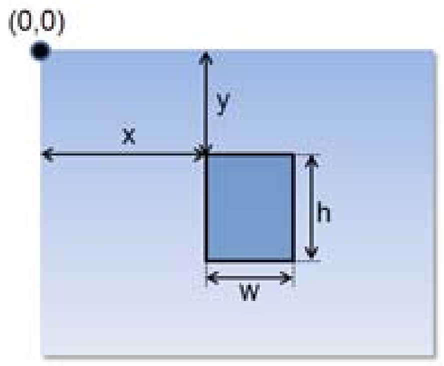


Figure 4.4: Parametrization of a region in an image.

Moreover, an additional parameter s_f can be added in order to flip between excitatory and inhibitory regions, obtaining complementary features (excitatory regions become inhibitory and viceversa). Using all the previous definitions, the parametrization for a Haar-like feature can be write as $\mathbf{s_F} = \{s_x, s_y, s_w, s_h, s_t, s_f\}$, where $s_x, s_y, s_w, s_h \in \mathbb{N}$, $s_t \in \{1, \ldots, 8\}$, and $s_f \in \{-1, 1\}$.

4.4.3 Final model

Once all the parameters involved on the weak hypothesis are described, we can define the parameter vector $\mathbf{s} = \mathbf{s_M} \cup \mathbf{s_F} = \{s_p, \mathbf{s_h}, s_x, s_y, s_w, s_h, s_t, s_f\}$. Since Haar-like features describe an image with a single value, $\mathbf{s_h}$ has dimension one and corresponds to a threshold value. Although all these parameters must be optimized in order to get a good weak hypothesis, not all of them must be learned using an evolutionary approach, but some of them can be either learned using other methods or fixed to a certain value. For instance, the threshold value of the classifier can be exhaustively found once the feature parameters have been learned with an evolutionary approach.

At this point, a simplification respect to the Viola & Jones approach is applied to our method. This simplification is to use threshold value fixed to zero, what is relate in the literature as ordinal features. The term of ordinal features is related to the use of the sign instead of directly the value of the feature. In [TS01], a face detection approach is presented using only the sign of region intensity differences, and their authors demonstrate that removing the magnitude of the difference, the model becomes more stable to illumination changes and image degradation. With this approach, we can remove the threshold value from the parameters vector. Moreover, using ordinal features, it is easy to verify that s_p and s_f have the same effect on the classification value, and therefore, only one of them must be learnt.

Finally, using all previous considerations, the problem consists on finding the model parameters:

$$\mathbf{s} = \{s_p, s_x, s_y, s_w, s_h, s_t\} \tag{4.1}$$

which minimize the weighted error function of the AdaBoost, under the following constrains:

$$\begin{array}{l} s_p \in \{-1, 1\} \\ s_x, s_y \geq 0 \\ s_w, s_h > 0 \\ s_x + s_w < W \\ s_y + s_h < H \\ s_t \in \{1, \ldots, 8\} \end{array} \tag{4.2}$$

where W and H corresponds to the width and height of the learning window.

4.5 Object detection based on Genetic Algorithms

Once the problem is formulated, in this section the implementation of an evolutionary weak learner based on Genetic Algorithms is presented. The first step is to define a chromosome based representation for problem variables in Eq. 4.1. Once the encoding is discussed, the adaption of a certain individual restricted to the constrains in Eq. 4.1

must be included in the evaluation function. Finally, an experiment to verify if the resulting schemata is able to learn is presented.

4.5.1 Chromosome encoding

Although the encoding of a problem in a chromosome-like representation can be performed in multiple ways, the use of binary representation simplifies mutation and cross-over operators, and it is the recommended in most problems. Since all our variables are integers, their representation do not need to store decimal positions. In addition, the ranges for each variable are known, therefore, we can adjust the number of bits in order to minimize values out of the valid ranges.

As general rule, the number of bits in order to represent a parameter x which take values into a range $[MinVal, \ldots, MaxVal]$, can be calculated as:

$$NumBits = \lceil \log_2 (MaxValue - MinValue + 1) \rceil \tag{4.3}$$

in the case of variables that take values from a set of non-contiguous values, we need to define a range of continuous values and map each value from the original values set to one of the values in the defined contiguous range.

In order to codify the parameter, we need to move it from its original range to a zero based range $\{0, \ldots, MaxVal - MinVal\}$ and codify the resulting number in a binary representation. Analogously, when we need to recover the value of a certain parameter, we need to decode the value codified in a binary format to their decimal representation and move this value to their original range.

Using the representation above, we need to differentiate between those parameters which depends on the problem, and therefore, for which we need problem information to decode, from those which are independent. Basically, the only problem information we need is the size of the training window, in order to adjust the range of region parameters (see Fig. 4.4). Assuming a training window of $W \times H$ pixels, the ranges and number of bits in order to represent each parameter is shown in Table 4.1.

Table 4.1
NUMBER OF BITS TO REPRESENT EACH PARAMETER.

Parameter Name	**Initial Range**	**Final Range**	**Number of Bits**
s_p	$\{-1, +1\}$	$[0_{-1}, 1_{+1}]$	1
s_x	$[0, W-1]$	$[0, W-1]$	$\lceil \log_2 W \rceil$
s_y	$[0, H-1]$	$[0, H-1]$	$\lceil \log_2 H \rceil$
s_w	$[0, W-1]$	$[0, W-1]$	$\lceil \log_2 W \rceil$
s_h	$[0, H-1]$	$[0, H-1]$	$\lceil \log_2 H \rceil$
s_t	$[1, 8]$	$[0, 7]$	3

The final chromosome representation for an individual is shown in Fig. 4.5, where the width of each field in bits and the final number of bits is represented. The last consideration in order to get the final representation, is how we codify a decimal number to their binary representation. We adopt a codification in terms of Gray codes, which guarantee that similar representations corresponds to similar values, although a direct binary representation could be used instead.

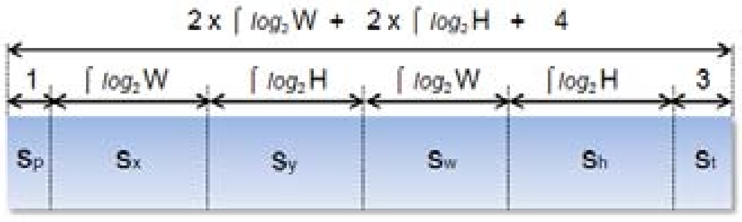


Figure 4.5: Final chromosome representation for an ordinal Haar-like feature based weak learner. The number of bits are calculated over a learning window of $W \times H$ pixels.

4.5.2 Evaluation function

The goal of the evaluation function is to assign an adaption value to each chromosome. The first step is to decode the chromosome into the original parameters vector, and to evaluate whether those parameters fulfill the constrains or not. When one individual do not fulfill some of the problem constrains, we have two possibilities:

Fixed Error Value: Since we are working on a minimization problem, we can define a value greater than the worse possible value than an individual can achieve. Therefore, in the evolutionary process, the individual will have near null possibilities to be selected for next generations.

Informative Error Value: The idea is similar to the fixed error value approach, but in this case, instead of returning a fixed value, a value according on how near is the individual to fulfill the constrains is calculated, therefore, if in some generation of the evolutionary process a large number of bad individuals is present, it is possible to evolve to a good individual. If a fixed value is used, no difference between bad individuals is provided, disallowing an evolution to a good individual.

Since in our case the range are adjusted to the possible values, it is more probable to have a large amount of valid individuals, and a fixed error value is used in order to obtain a faster evaluation function.

For valid individuals, the weighted error function is calculated using a training set and the weights associated to each sample. This process is summarized in Algorithm 21. Note that that a function $f_{\mathbf{s}}$ must be defined, which calculate the value of the Haar-like feature. This process can be done using an image representation based on integral images (see Section 2.7.3). In this implementation, the polarity value s_p is the classifier polarity, therefore, $f_{\mathbf{s}}$ do not consider a polarity value and it is applied over the final hypothesis. Note also that a fixed error value of 1.1 is used to indicate when the individual do not fulfill the constrains. Since the worse possible error value is 1.0, this assigned value is assumed to be always worse than the maximum error value for a valid classifier.

Algorithm 21 Evaluation function for ordinal Haar-like feature chromosome based representation.

Input: A training set X of N pairs $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ is the i^{th} image and $y_i \in \{-1, 1\}$ is the category of the object present in $\mathbf{x}_i$, a weights distribution $W = \{w_1, \ldots, w_N\}$ where w_i is the weight of i^{th} sample in X, the chromosome C to be evaluated, and $f_{\mathbf{s}}(x) \mapsto \mathbb{Z}$ that calculates the value of the Haar-like feature parameterized in $\mathbf{s} = \{s_x, s_y, s_w, s_h, s_t\}$ over a given image $\mathbf{x}$.

```
Decode the chromosome $C$ into the parameters vector $\mathbf{s} = \{s_p, s_x, s_y, s_w, s_h, s_t\}$.
if $\mathbf{s}$ fulfill the constrains of the problem then
  Initialize $\varepsilon \leftarrow 0$
  for $i = 1, .., N$ do
    if $f_{\mathbf{s}}(x_i) \geq 0$ then
      $h_i \leftarrow +1$
    else
      $h_i \leftarrow -1$
    end if
    $h_i \leftarrow h_i \times s_p$
    if $h_i \neq y_i$ then
      $\varepsilon \leftarrow \varepsilon + w_i$
    end if
  end for
else
  $\varepsilon \leftarrow 1.1$
end if
```

Output: the final error value ε

4.5.3 Results

Once codification and evaluation function have been defined, we can use a Genetic Algorithm in order to learn a classifier from a training set. In our experiments, the implementation of Genetic Algorithms in the Matlab function optimization toolbox is used. As introduced above, the evolutionary algorithm is used as the weak learner process in the AdaBoost algorithm. At this point, we just need to define the parameters of the Genetic Algorithm in order to use an evolutionary learning algorithm in our tests.

Experimental setting

Using the concepts defined in Section 3.2, we define a population size of 100 individuals, a Gaussian based mutation probability (the Gaussian is centered at zero with a variance of the half of the variable range, decreasing the variance along the generations), and scattered cross-over strategy with a cross-over fraction of 0.8. Although the defined evolutionary weak learner can be used in any AdaBoost variant, in our experiment we use Gentle AdaBoost defined in Section 2.6.2.

Testing the Learning capabilities

Before continuing exploring evolutionary methodologies to improve object detection methods, we need to verify if the use of those methodologies combined with the classical approach is able to learn, and in affirmative case, how it learn in comparison with the classical approach.

To compare the Evolutionary AdaBoost with their exhaustive version, we need to define a problem where both can be applied, that is, a problem where the cardinality of the search space is enough small to allows exhaustive searches. For this purpose, we use the face detection database (Section C.1), where the images are small enough to allow exhaustive searches. The images have been divided in two different balanced groups, the first one used to learn and the second one to test the learned classifier. These sets remain fixed during all the experiment, allowing to compare the evolution of the learning process in both approaches.

When we use a Genetic Algorithm instead of an exhaustive search, different initializations of the algorithm with the same training data give rise to different weak classifiers. An one-stage detector is learnt using the training and test sets, comparing the error evolution for both strategies, and the variance in the evolutive approach over different runs. The learning process is repeated 50 times, using 50 iterations of the Evolutionary Adaboost. In the case of the classic Adaboost, as the *Weak Learner* does an exhaustive search over the features, at each round the selected features are the same. At the end, for the evolutionary approach we calculate the mean error value and the variance over all the rounds for each iteration.

In Figure 4.6 the train and test mean error values at each iteration of the AdaBoost are shown. Note that both methods converge with the same number of iterations and have a similar behavior. Moreover, we can see that in some iterations, the evolutionary approach outperform the exhaustive approach. This can be explained because AdaBoost approach only ensure that using the error descendant approach, the learning process converges, and a feature is selected only taking into account previously selected features, not the total set of features. That is, it exists better sets of features that solves the problem, but find the optimum combination is unfeasible. Using AdaBoost we obtain a good approximation to the optimum combination at a reasonable time, but it is possible to find better solutions.

To analyze the effect of randomness in the evolutionary version, the error variability during the learning process is shown in Fig. 4.7 we show the mean and standard deviation for the error at each iteration. The confidence interval shows that the variance is very small. Therefore, though the evolutive Adaboost has a random component, the goodness of the given solution is similar. Moreover, the variability on the error decreases along the iterations.

Testing the model versatility

Once we see that the evolutionary algorithm is able to learn a classifier with a similar convergence than the exhaustive algorithm, the versatility of the evolutionary approach over different object detection problems is tested. We choose a set of five

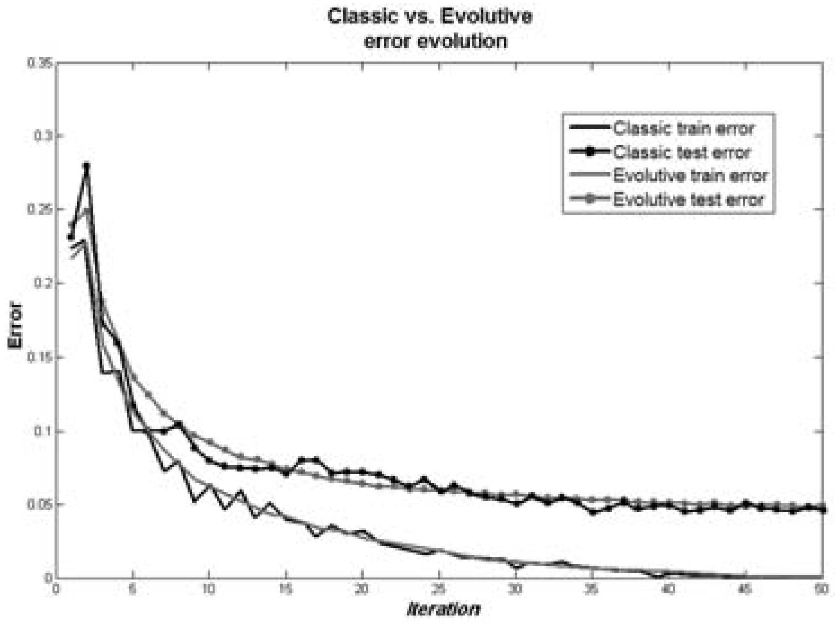


Figure 4.6: Error evolution using the classic Adaboost approach and the genetic WeakLearner

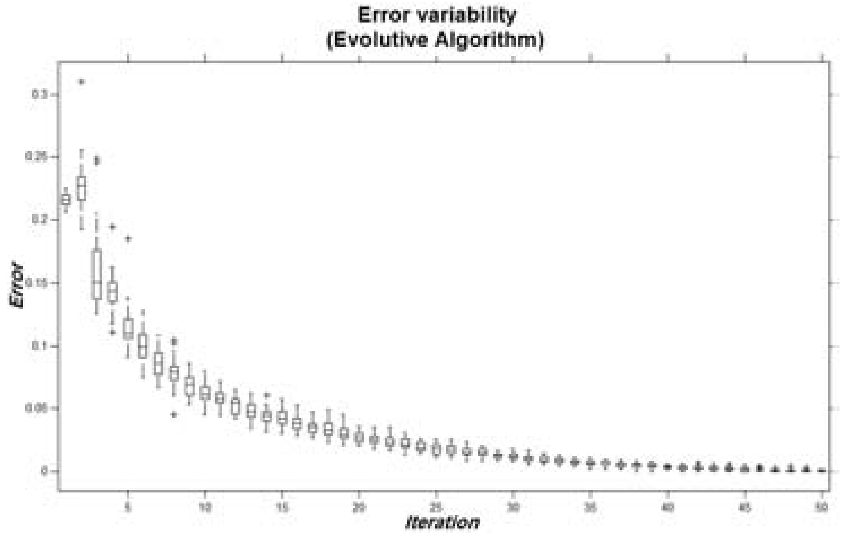


Figure 4.7: Genetic approach. Error variability on the training process.

different problems, four correspond to public datasets and one using the traffic sign dataset obtained for the Geomobil project (see Appendix A). Some examples of positive samples for each dataset are shown in Fig. 4.8, and a more detailed description is provided in Appendix C.

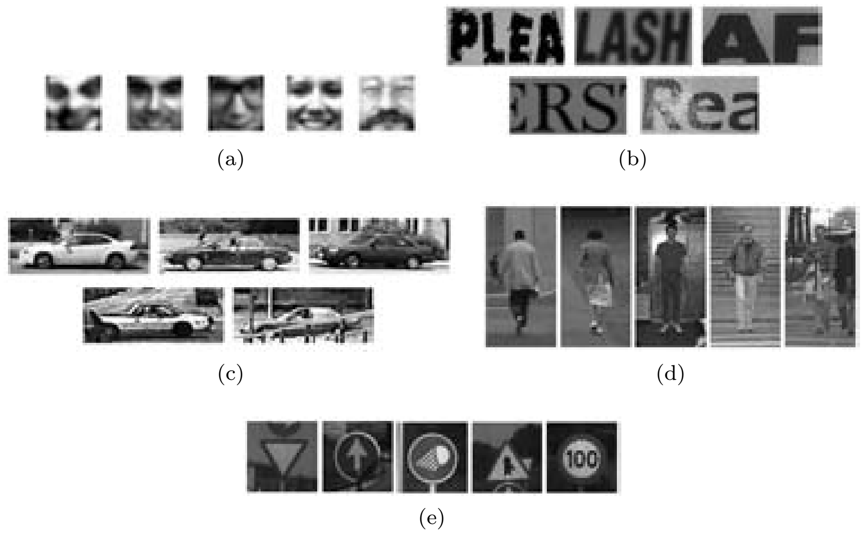


Figure 4.8: Data set examples. *a)* Faces *b)* Text *c)* Cars *d)* Pedestrians *e)* Traffic signs

Tests are done using a stratified ten-fold strategy with these problems, which consists of dividing the samples in the data set into ten balanced disjoint subsets, and perform ten experiments using one subset as test data and the remaining nine subsets as learning data. At the end, the final result corresponds to the mean of the ten results and the confidence interval at 95%. The number of iteration of the AdaBoost for each dataset is 200 and the number of individuals in the Genetic Algorithm is 100. The final performance and the balanced error for the different data sets is shown in Table 4.2.

Table 4.2
Performance and Balanced Error for the Evolutionary AdaBoost using Haar-like features and Genetic Algorithms.

Data set	**Performance**	**BER**
Cars	71.52% ± 7.53	29.29% ± 7.62
Faces	59.10% ± 1.99	43.50% ± 4.39
Text	48.60% ± 3.75	51.83% ± 5.41
Pedestrians	52.29% ± 5.93	44.05% ± 6.19
Traffic Signs	56.60% ± 2.85	34.50% ± 4.10

Although the errors for some problem are relatively large, in this experiments we use only one classifier, and the error values can be improved using a cascade of classifiers. Nevertheless, the use of a cascade of classifiers difficults a clear comparison between methodologies. The values in Table 4.2 provide reference values to compare with future approaches. Moreover, using a single classifier instead of a cascade of classifiers allows us to calculate the area under the ROC curve (AUC), which is demonstrated to be the better measure to compare different detectors. The area under ROC curve is equivalent to the probability that a randomly chosen negative example will have a smaller estimated probability of belonging to the positive class than a randomly chosen positive example. In [HL05] a comparison between AUC and accuracy measures to the evaluation of learning algorithm concludes that AUC is a better measure. Hand and Till [HT01] present the following simple approach to calculating AUC of a classifier for binary classification:

$$\hat{A} = \frac{S_0 - n_0(n_0+1)/2}{n_0 n_1} \tag{4.4}$$

where n_0 and n_1 are the numbers of positive and negative examples, respectively, and $S_0 = \sum r_i$, where r_i is the rank of the i^{th} positive example in the ranked list.

In order to build the ranked list from our learned detectors, we reformulate the output of the Gentle Adaboost as:

$$H'(x) = \sum_{t=1}^{T} h_t(x) \tag{4.5}$$

with this new formulation, the rank r_i can be calculated as the number of elements within the set R_i defined as:

$$R_i = \{x_j | \forall j \quad y_j = -1, H'(x_j) < H'(x_i)\} \tag{4.6}$$

Using this formulation, results in Table 4.2 can be written in terms of their AUC (see Table 4.3), and will be considered as reference values in future experiments. In Fig. 4.9 and 4.10 some false positive and false negative images are shown for each problem. In the case of false negative samples, homogeneous regions are often selected as true objects because of noise. In these cases variance filters can be applied to remove these images, improving the results.

Table 4.3
Area under the ROC curve for the Evolutionary AdaBoost using Haar-like features and Genetic Algorithms.

Data set	**AUC**
Cars	$69.65\% \pm 7.54$
Faces	$55.22\% \pm 4.23$
Text	$45.84\% \pm 5.75$
Pedestrians	$54.00\% \pm 6.68$
Traffic Signs	$63.30\% \pm 4.41$

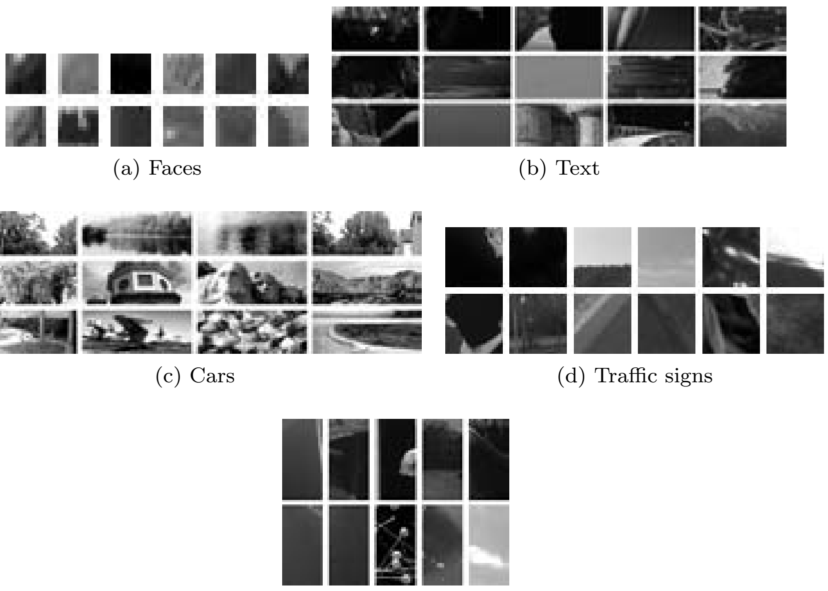


Figure 4.9: False Positives for the Evolutionary AdaBoost using Haar-like features and Genetic Algorithms.

4.6 Extending the feature set

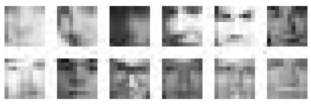
Once verified the learning ability of the evolutionary AdaBoost, we have a framework where the cardinality of the feature space is no more a restriction, and therefore, it is possible to explore feature sets that in the classical AdaBoost framework are unfeasible. This is the case of Dissociated Dipoles introduced by Shina [BS03] (see Section 2.3.2), another type of features based on rectangular regions subtraction which allows non local comparisons.

4.6.1 Dissociated Dipoles

In order to use Dissociated Dipoles in the evolutionary AdaBoost framework, it is necessary to parameterize this feature family. In contrast with Haar-like features, Dissociated Dipoles do not follow predefined configurations, but their two regions can have arbitrary positions and sizes. Using the same representation for a region that in the case of Haar-like features, we can parameterize the Dissociated Dipoles using their regions parameters (see Fig. 4.11):

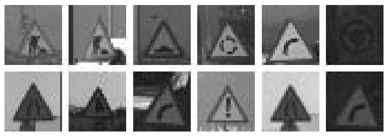
$$\mathbf{s_F} = \{s_{x_e}, s_{y_e}, s_{w_e}, s_{h_e}, s_{x_i}, s_{y_i}, s_{w_i}, s_{h_i}, s_f\} \tag{4.7}$$

where $s_{x_{e,i}}, s_{y_{e,i}}, s_{w_{e,i}}, s_{h_{e,i}} \in \mathbb{N}$ and $s_f \in \{-1, 1\}$ is a flag to interchange excitatory and inhibitory regions.


(a) Faces

(b) Text

(c) Cars


(d) Traffic signs

(e) Pedestrians

Figure 4.10: False Negatives for the Evolutionary AdaBoost using Haar-like features and Genetic Algorithms.

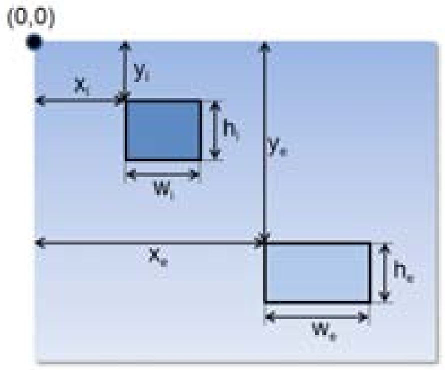


Figure 4.11: Parametrization of both regions of a Dissociated Dipole in an image. Lighter region corresponds to the excitatory pole and darker region to the inhibitory pole.

Using the same classification scheme than in the case of Haar-like features, we can use the same simplification for the flipping parameter and polarity parameter. Finally, the problem consists on finding the model parameters:

$$\mathbf{s} = \{s_p, s_{x_e}, s_{y_e}, s_{w_e}, s_{h_e}, s_{x_i}, s_{y_i}, s_{w_i}, s_{h_i}\} \tag{4.8}$$

which minimize the weighted error function of the AdaBoost, subject to the following constrains:

$$\begin{aligned} &s_p \in \{-1, 1\} \\ &s_{x_{e,i}}, s_{y_{e,i}} \geq 0 \\ &s_{w_{e,i}}, s_{h_{e,i}} > 0 \\ &s_{x_e} + s_{w_e} < W \\ &s_{y_e} + s_{h_e} < H \\ &s_{x_i} + s_{w_i} < W \\ &s_{y_i} + s_{h_i} < H \end{aligned} \tag{4.9}$$

The chromosome-like codification of the parameters is performed in the same way that in the case of Haar-like features, obtaining the chromosome shown in Fig. 4.12.

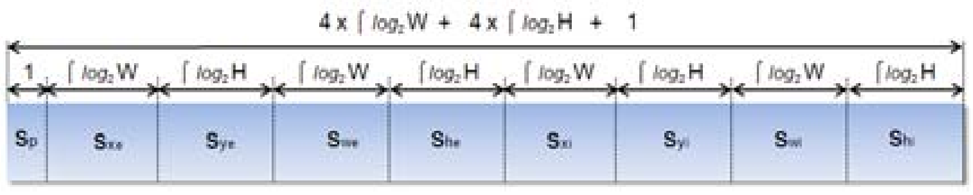


Figure 4.12: Final chromosome representation for an ordinal Dissociated dipole feature based weak learner. The number of bits are calculated over a learning window of $W \times H$ pixels.

To test the effectiveness of dissociated dipoles, we repeat the experiment detailed in Section 4.5.3 using the evolutionary AdaBoost with a weak learner that use ordinal Dissociated Dipoles. The results are shown in Table 4.4, where the ranking for each method is shown in brackets ignoring the confidence interval as is required for posterior statistical analysis.

Table 4.4
AREA UNDER THE ROC CURVE FOR THE EVOLUTIONARY ADABOOST USING HAAR-LIKE FEATURES AND DISSOCIATED DIPOLES. LEARNING PROCESS IS PERFORMED BY AN EVOLUTIONARY ADABOOST WITH GENETIC ALGORITHMS AS WEAK LEARNER.

	AUC	
Data set	***Haar-like***	***Dissociated Dipoles***
Cars	$69.65\% \pm 7.54(2)$	$79.27\% \pm 5.64(1)$
Faces	$55.22\% \pm 4.23(2)$	$67.47\% \pm 4.52(1)$
Text	$45.84\% \pm 5.75(2)$	$50.33\% \pm 8.75(1)$
Pedestrians	$54.00\% \pm 6.68(2)$	$68.16\% \pm 5.57(1)$
Traffic Signs	$63.30\% \pm 4.41(1)$	$61.53\% \pm 7.17(2)$
Mean Rank	1.8	1.2

Using the statistical analysis developed by Demšar in [Dem06] (see Appendix D) we study the statistical significance of the results. In this study the combination of a feature set and the evolutive Adaboost is referred as algorithm.

Let r_i^j be the rank of the j-th of k algorithms on the i-th of N data sets. The Friedman test compares the average ranks of algorithms, $R_j = \frac{1}{N}\sum_i r_i^j$. Under the null-hypothesis, which states that all the feature sets are equivalent, and so their average ranks R_j are equal, the Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] = 1.8 \tag{4.10}$$

is distributed according to χ_F^2 with $k-1$ degrees of freedom when N and k are big enough. For a small number of algorithms and data sets, exact critical values have been computed. Following the considerations of Iman and Davenport [ID80], we compute:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2} = \frac{4\times\chi_F^2}{5-\chi_F^2} = 2.25 \tag{4.11}$$

which is distributed according to the F-distribution with $k-1=1$ and $(k-1)(N-1) = 1\times 4 = 4$ degrees of freedom. The critical value of $F(1,4)$ for $\alpha = 0.05$ is 7.71, which is larger than F_F, so we cannot reject the null-hypothesis at 95%. The critical value of $F(1,4)$ for $\alpha = 0.1$ is 4.54, even larger than F_F, therefore, null-hypothesis cannot be rejected at 90%.

Although Dissociated Dipoles perform better in most of the experiments, the result of the statistical analysis reflects that with our data, we cannot state that it is a statistically significant difference between both features sets.

In contrast with the initial assumption derived from the work of Lienhart & Maydt [LM02] that a large feature set not only improve the results but also the convergence of AdaBoost algorithm, we find that Dissociated Dipoles which has a cardinality much larger than Haar-like features do not perform statistically better than Haar-like features.

Analyzing both feature sets, most relevant benefits of each feature set are highlighted in Table 4.5.

Table 4.5
Feature sets comparison. Benefits of Haar-like features and Dissociated Dipoles.

Shared	***Haar-like***	***Dissociated Dipoles***
◇ Illuminance invariance	◇ Line detectors	◇ Non-local comparisons
◇ Fast computation	◇ Center-surround analysis	◇ Large cardinality
◇ Easily scalable for multi-scale detection.		
◇ Edge detector		

From the list of benefits, we can state that most of benefits are shared, and the most interesting differences are the ability of Haar-like features to detect lines and center-surrounding structures. Therefore, if we can extend Dissociated Dipoles to incorporate these interesting Haar-like configurations, we could obtain a new feature set that shares the benefits of Haar-like and Dissociated dipoles. In next section, we propose a new feature which attempt to perform this fusion.

4.6.2 Weighted Dissociated Dipoles

Weighted Dissociated Dipoles appear from the wish to build a feature set which shares benefits form Haar-like features and Dissociated Dipoles. The extension proposed is to add a weight $\alpha \in \{1, 2\}$ to each of two poles of the Dissociated Dipole. This weight is applied to the value of the pole region, and allows to remove overlapped regions that in other way are compensated. An example of how a Haar-like line detector is approximated using the Weighted Dissociated Dipoles is shown in Fig. 4.13.

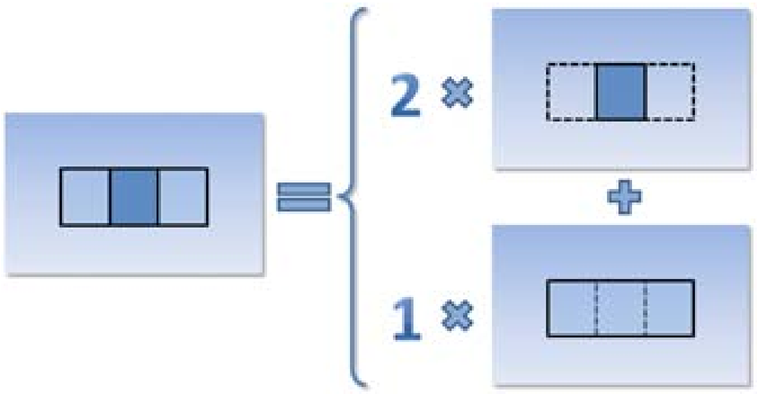


Figure 4.13: A Haar-like feature can be approximated using weights on the dissociated dipoles. In this example, a vertical detector is approximated using a weight two for the inhibitory pole.

Following a similar formulation and the same classification scheme that in the case of Dissociated Dipoles, and using the same simplification for the flipping parameter and polarity parameter, the problem consists on finding the model parameters:

$$\mathbf{s} = \{s_p, s_{x_e}, s_{y_e}, s_{w_e}, s_{h_e}, s_{x_i}, s_{y_i}, s_{w_i}, s_{h_i}, s_{\alpha_e}, s_{\alpha_i}\} \tag{4.12}$$

which minimize the weighted error function of the AdaBoost, subject to the following constrains:

$$\begin{array}{l} s_p \in \{-1, 1\} \\ s_{x_{e,i}}, s_{y_{e,i}} \geq 0 \\ s_{w_{e,i}}, s_{h_{e,i}} > 0 \\ s_{x_e} + s_{w_e} < W \\ s_{y_e} + s_{h_e} < H \\ s_{x_i} + s_{w_i} < W \\ s_{y_i} + s_{h_i} < H \\ s_{\alpha_{e,i}} \in \{1, 2\} \end{array} \tag{4.13}$$

The chromosome-like codification of the parameters is performed in the same way that in previous cases, obtaining the chromosome shown in Fig. 4.14.

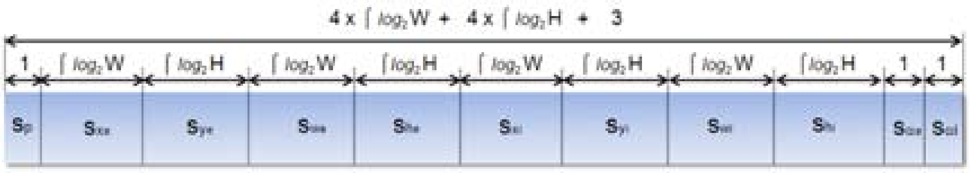


Figure 4.14: Final chromosome representation for an ordinal Weighted Dissociated dipole feature based weak learner. The number of bits are calculated over a learning window of $W \times H$ pixels.

If our hypothesis is correct, the non local information that incorporates Dissociated Dipoles was compensated by the special structures detection configurations present of Haar-like features. Using a feature set that shares benefits from both feature sets must improve the results. To test if this new feature set is better than previous ones, we repeat the experiment detailed in Section 4.5.3 using the evolutionary AdaBoost with a weak learner that use ordinal Weighted Dissociated Dipoles. The results are shown in Table 4.6, where the ranking for each method is shown in brackets ignoring the confidence interval as is required for posterior statistical analysis.

Table 4.6
AREA UNDER THE ROC CURVE FOR THE EVOLUTIONARY ADABOOST USING HAAR-LIKE FEATURES, DISSOCIATED DIPOLES, AND WEIGHTED DISSOCIATED DIPOLES. LEARNING PROCESS IS PERFORMED BY AN EVOLUTIONARY ADABOOST WITH GENETIC ALGORITHMS AS WEAK LEARNER.

	AUC		
	Haar-like	***Dissociated Dipoles***	***Weighted Diss. Dip.***
Cars	$69.65\% \pm 7.54(3)$	$79.27\% \pm 5.64(2)$	$95.21\% \pm 3.28(1)$
Faces	$55.22\% \pm 4.23(3)$	$67.47\% \pm 4.52(2)$	$87.74\% \pm 2.85(1)$
Text	$45.84\% \pm 5.75(3)$	$50.33\% \pm 8.75(2)$	$80.35\% \pm 5.08(1)$
Pedestrians	$54.00\% \pm 6.68(3)$	$68.16\% \pm 5.57(2)$	$88.40\% \pm 2.40(1)$
Traffic Signs	$63.30\% \pm 4.41(2)$	$61.53\% \pm 7.17(3)$	$87.92\% \pm 3.61(1)$
Mean Rank	2.8	2.2	1.0

Under the null-hypothesis, which states that all the feature sets are equivalent, and so their average ranks R_j are equal, the Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] = 8.4 \tag{4.14}$$

is distributed according to χ_F^2 with $k-1$ degrees of freedom when N and k are big enough. We compute Iman and Davenport derived statistic:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2} = \frac{4 \times \chi_F^2}{5-\chi_F^2} = 21.0 \tag{4.15}$$

which is distributed according to the F-distribution with $k-1=2$ and $(k-1)(N-1) = 2 \times 4 = 8$ degrees of freedom. The critical value of $F(2,8)$ for $\rho = 0.05$ is 4.45, which is lower than F_F, so we reject the null-hypothesis at 95%.

Once the null-hypothesis has been rejected, we know that algorithms are not statistically equivalent, therefore, we can proceed with a post-hoc test. In our case, as no algorithm is singled out for comparisons, we use the Nemenyi test for pairwise comparisons. The performance of the two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} = 1.48 \tag{4.16}$$

where critical values q_α are based on the Studentized range statistic divided by $\sqrt{2}$. Using averaged ranks in table 4.6 we calculate all the pair-wise differences. Comparing those differences with the critical value, we can conclude that the Weighted Dissociated Dipoles are significantly better than the Haar-like features ($2.8 - 1.0 = 1.8 > 1.48$), but can say nothing about the difference between Dissociated Dipoles and Weighted Dissociated Dipoles ($2.2 - 1.0 = 1.2 < 1.48$), and between Dissociated Dipoles and Haar-like features ($2.8 - 2.2 = 0.6 < 1.48$). Using $\rho = 0.1$, we obtain $CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} = 1.30$ which produce the same results that for $\rho = 0.05$. Those results are graphically represented in Fig. 4.15.

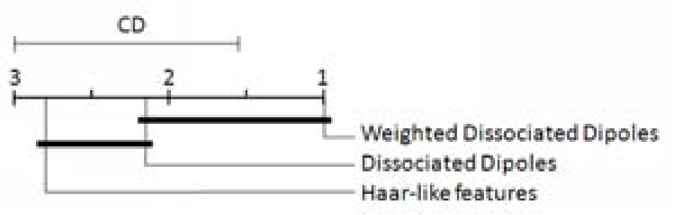


Figure 4.15: Comparison of all feature sets using AdaBoost and Genetic Algorithms against each other with the Nemenyi test. Groups of features that are not significantly different (at $\rho = 0.05$) are connected.

As a conclusion, adding non local comparisons to Haar-like structure detectors, we obtain a statistically significant improvement on the results. Therefore, the classical approach can be improved using richer feature sets, which often means increase the cardinality of the initial feature set. In this section, the standard evolutionary approach, that is, Genetic Algorithms has been used to verify that efforts to break AdaBoost limitations in the cardinality of used features has sense and improve the results.

After features improvement and the justification of evolutionary methods on the AdaBoost algorithm, next sections in this chapter are related to the improvement of the learning methodology.

4.7 Object detection based on Probabilistic Darwin Machines

Genetic Algorithms are the classical approach to optimization problems, and they are used as reference evolutionary strategy in close all the works where evolutionary computation is involved. However, efforts of researchers in this field caused the apparition of a large variety of new evolutionary methods, which improve the canonical Genetic Algorithm is most aspects. One of the newest paradigms on the evolutionary computation field, and which demonstrated to outperform Genetic Algorithms in optimization problems are the Probabilistic Darwin Machines introduced in Section 3.3. Comparatives performed by Larrañaga et al. [LL02, LLM03, LLIB06] demonstrated that those new algorithms improve the results and in most cases the convergence of the search.

The use of Probabilistic Darwin Machines in our work is motivated not only by the potential improvement on the convergence and quality of the results, but for the additional information contained on the final probability distributions. The use of probabilistic models allow to find relationships between parameters, an estimation of optimal regions in the search space, and open the door to the use of prior models to guide the search.

4.7.1 Problem redefinition

Moving from Genetic Algorithms to Probabilistic Darwin Machines represents some slightly changes in the definition of the problem, mainly reflected in the manner individuals are represented and how they are evaluated. Since evolution is based on probability models estimation, to select a good probability model which correctly adapts to the problem is a key point in the method definition. In the following these modifications are discussed.

Individuals encoding

In contrast with Genetic Algorithms, in the case of Probabilistic Darwin Machines, individuals do not need to be encoded into a chromosome-like codification, and the problem parameters generally are not binary. The codification of an individual is performed assuming each parameter as a random variable, which probability distribution is estimated among generations.

Evaluation function

The evaluation function in a Probabilistic Darwin Machine is equivalent to the one presented for the Genetic Algorithms (see Section 4.5.2), because this is one of the problem dependant parts of an evolutionary algorithm. However, individuals in a population are the values for the parameters, therefore, no decoding step in necessary.

Probability model selection

One of the most critical decisions when Probability Darwin Machines are used is which probability model will be used during the evolutionary process. There are basically two main aspects to take into account to select the probability model:

- A problem can be defined by discrete random variables or continuous ones. Moreover, in a same problem, some variables can take discrete values while other variables are continuous. Therefore, the selected model must allow the estimation of all the types of variables in the problem
- The different variables of the problem can be correlated, in which case, a model sensitive to the interactions present on the problem will perform better than a simpler model. However, using a more complex model usually needs large amount of data and the learning time can increase. We need to choose the simplest model which allows to represent variable interactions.

4.7.2 Results

In order to compare the use of Probabilistic Darwin Machines on the evolutionary Adaboost framework, the experiments performed with Genetic Algorithms are repeated changing only the evolutionary algorithm. To allow comparisons, we maintain either the number of iterations for the AdaBoost algorithm at 200 and the number of individuals at 100. The definitions for problem parameters are the same used with the Genetic algorithms, and defined in Equations 4.1, 4.8, and 4.12, with their related constrains defined in Equations 4.2, 4.9, and 4.13. Moreover, the evaluation function defined in Algorithm 21 is used in these experiments, with the exception that in this case the input is not a chromosome, but directly the values, and therefore, the decoding process is not used anymore.

If we analyze the random variables, it is easy to discover that they are not independent. For instance, regions with positions near the right or bottom sides of the training window cannot have large sizes, because the codified feature becomes invalid. At the same time, when using Haar-like features, for a given position and size, some configurations can invalidate the feature, which suggest multi-variate dependencies. In Dissociated Dipoles and Weighted dissociated dipoles, the overlapping degree between regions influences the goodness of a features (i.e. the output for two totally overlapped poles is always zero), obtaining a dependency between parameters of both dipoles. Those observations suggested the use of probability models with the ability of represent multiple dependencies between variables, discarding the use of univariate and bivariate models.

The simplest model which allows to learn multiple dependencies is the Extended Compact Genetic Algorithm (EcGA), introduced in Section 3.3.4. This method creates cluster of dependant variables, estimating a probability model according to these dependencies. The results obtained using EcGA in the different datasets using the experimental setting defined in 4.5.3 are presented in Table 4.7. For those experiments, we use the code provided by Lobo et al. [LSH06].

Note that the obtained ranks are the same than in the case of Genetic Algorithms, therefore, if we apply the statistical analysis performed in last section for the data in

Table 4.7
Area under the ROC curve for the Evolutionary AdaBoost using Haar-like features, Dissociated Dipoles, and Weighted Dissociated Dipoles. Learning process is performed by an evolutionary AdaBoost with Extended compact Genetic Algorithms (EcGA) as weak learner.

	AUC		
	Haar-like	***Dissociated Dipoles***	***Weighted Diss. Dip.***
Cars	73.80% ± 6.49(3)	76.19% ± 6.70(2)	85.99% ± 6.72(1)
Faces	52.96% ± 7.17(3)	68.63% ± 7.48(2)	79.73% ± 6.84(1)
Text	47.83% ± 3.23(3)	50.80% ± 6.48(2)	79.88% ± 7.15(1)
Pedestrians	55.39% ± 5.54(3)	68.22% ± 8.83(2)	80.55% ± 8.88(1)
Traffic Signs	64.60% ± 4.78(2)	63.13% ± 6.52(3)	89.71% ± 2.38(1)
Mean Rank	2.8	2.2	1.0

Table 4.6, we will obtain the same results, that is, the description power of the features sets remains the same despite the use of different learning algorithms. In order to compare the learning algorithms, in Table 4.8 the results of EcGA are compared with the obtained with GA, considering each combination of dataset and feature set as an experiment, and both evolutionary algorithms as methods. For comparison purposes, the mean rank for each feature is evaluated separately from the overall rank.

Following Demšar statistical comparison methodology [Dem06], all the statistics are computed and summarized in Table 4.9. Although EcGA performs better in most of the experiments, the test shows that this difference is not statistically significant, neither for individual features and the overall performance. Moreover, in the results in Table 4.8 we can observe that EcGA performs better for Haar-like features and Dissociated Dipoles, but with the Weighted Dissociated Dipoles GA obtain better results. Our hypothesis is that EcGA has not enough description power to deal with the new dependencies introduced with the weights, and therefore, some of those relationships are not represented in the learnt model.

4.8 PDM based on Naïve Bayes models Estimation

The use of EcGA as Weak Learner has demonstrated that the performance of the standard Genetic Algorithms estimating the parameters of features can be improved in some cases by using Probability Darwin Machines. Although the probability model used in EcGA is able to create clusters of variables with dependencies, allowing to describe multiple dependencies among some problem variables, this structure is limited to these learned clusters, which is reflected in the poor results obtained when we face more complex models, such as Weighted Dissociated Dipoles.

The use of more powerful probability model can allow the use of more complex dependencies, obtaining a model with a better adaptation to each problem. In the other hand, more complex models can require complex estimation algorithms which needs larger amount of data and increase the learning time. From the point of de-

Table 4.8
Area under the ROC curve for the Evolutionary AdaBoost using Haar-like features, Dissociated Dipoles, and Weighted Dissociated Dipoles. Comparative between Genetic Algorithms (GA) and Extended compact Genetic Algorithms (EcGA) as weak learners.

Experiment		**Method**	
Features	**Dataset**	**GA**	**EcGA**
Haar-like	Cars	69.65%(2)	73.80%(1)
	Faces	55.22%(1)	52.96%(2)
	Text	45.84%(2)	47.83%(1)
	Pedestrians	54.00%(2)	55.39%(1)
	Traffic Signs	63.30%(2)	64.60%(1)
Mean Rank		1.8	1.2
Dis. Dip	Cars	79.27%(1)	76.19%(2)
	Faces	67.47%(2)	68.63%(1)
	Text	50.33%(2)	50.80%(1)
	Pedestrians	68.16%(2)	68.22%(1)
	Traffic Signs	61.53%(2)	63.13%(1)
Mean Rank		1.8	1.2
W.D.D.	Cars	95.21%(1)	85.99%(2)
	Faces	87.74%(1)	79.73%(2)
	Text	80.35%(1)	79.88%(2)
	Pedestrians	88.40%(1)	80.55%(2)
	Traffic Signs	87.92%(2)	89.71%(1)
Mean Rank		1.2	1.8
Mean Rank		1.6	1.4

Table 4.9
Statistics and critical values for experiments in Table 4.8.

			CV $\rho = 0.05$		**CV $\rho = 0.1$**	
Features	χ^2_F	F_F	χ^2_F	F_F	χ^2_F	F_F
Haar-like	1.8	2.25	3.84	7.71	2.71	4.54
Dis. Dip	1.8	2.25	3.84	7.71	2.71	4.54
W.D.D.	1.8	2.25	3.84	7.71	2.71	4.54
All	0.6	0.58	3.84	4.6	2.71	3.1

scription power, the Bayesian Network (or a belief network) have demonstrated to be the better solution, and is widely used in the literature.

Given a set of random variables $\{X_1, X_2, ..., X_N\}$, representing a Bayesian network using probabilistic graphical models is done using an acyclic directed graph, where each node corresponds to a variable (measured parameter, latent variable or hypothesis) and whose arcs encode the dependence between variables (see Fig. 4.16). Denoting as π_i the set of parents of X_i (nodes with an arc pointing to X_i), and as-

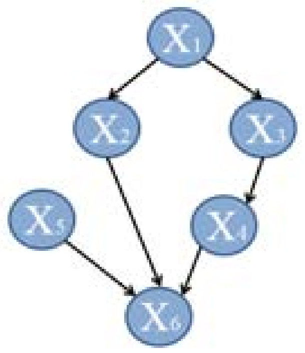


Figure 4.16: Bayesian Network representing a set of 6 random variables $\{X_1, ..., X_6\}$ and the dependencies between them.

suming that each node is conditionally independent of its non-descendants given its parents, the joint probability distribution can be conveniently written as the product of the local distributions of each node and its parents:

$$Pr(X_i, ..., X_N) = \prod_{i=1}^{N} Pr(X_i|\pi_i) \tag{4.17}$$

Learning a Bayesian network from data is a two-fold problem: Structure learning and parameter estimation. Although there exist good methods to estimate the structure and parameters of a Bayesian network, because exact inference is $\#P$-complete and thus the existent methods are often too costly, approximate methods like Markov Chain Monte Carlo [GRS96] and loopy belief propagation [YFW] must be used. The applicability of the Bayesian networks is limited by the fact that these methods have an unpredictable inference time and its convergence is difficult to diagnose.

In [LD05] Lowd & Domingos demonstrated from an empirical point of view that the representation power of a Bayesian network can be approximated using Naïve Bayes models, and presented an optimal algorithm to estimate those models. In the following, a Probabilistic Darwin Machine based on Lowd & Domingos estimation algorithm is presented. Since the estimation algorithm was so-called *Naïve Bayes models Estimation* (NBE) by their authors, we refer to this new evolutionary algorithm *Probability Darwin Machine based on Naïve Bayes models Estimation* (PDMNBE).

4.8.1 Naïve Bayes Models

The "naive" assumption that all variables are mutually independent given a "special" variable C, Bayesian networks are simplified (see Fig. 4.17), and the joint probability distribution is then given compactly by:

$$Pr(C, X_i, ..., X_N) = Pr(C) \prod_{i=1}^{N} Pr(X_i|C) \tag{4.18}$$

where the univariate conditional distributions $Pr(X_i|C)$ can take any form (e.g., multinomial for discrete variables, Gaussian for continuous ones). Naïve Bayes models allows very efficient inference of marginal distributions:

$$Pr(X = x) = \sum_{x=1}^{k} Pr(c) \prod_{i=1}^{|X|} Pr(X_i|c) \tag{4.19}$$

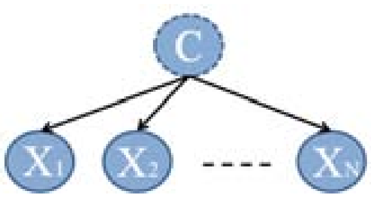


Figure 4.17: Naïve Bayes Model for a set of N random variables $\{X_1, ..., X_N\}$ and the hidden discrete variable C. Given the value of C, we can assume an univariate model over the rest of the variables.

4.8.2 Model Estimation

The estimation of Naïve Bayes model is performed using the Naïve Bayes models Estimation (NBE) algorithm proposed by Lowd & Domingos [LD05], which consists of an Expectation Maximization (EM) wrapped in an outer loop that progressively adds and prunes mixture components (see Alg. 22). The input data is split into a training set and a hold-out set. It begins with a single component consisting of each variable's marginal distribution. At each cycle, k new components are added, using a random training example to initialize each component, and removing these k seed examples from the data to avoid overfitting. The number of components k is doubled at each cycle. If there are m components before the cycle starts and n new ones are added, the weight $Pr(c)$ of each pre-existing component is rescaled by $m/(m+n)$, and each new component receives an initial weight of $1/(m+n)$.

Within each cycle, until the log likelihood of the hold-out data fails to increase by at least a fraction δ_{EM}, the expanded set of components is fitted using EM. At each iteration, the current model is saved if it yields the best holdout log likelihood so far. Since each step of EM takes time linear in the number of components, every five EM steps and after it ends, the low-weight components are pruned out in order to speed up the learning process. When an entire refinement step passes with little (less than δ_{Add}) or no improvement on the hold-out set, two final steps of EM on the best model are done with all the data.

4.8.3 Model Sampling

In the Probabilistic Darwin Machine framework, once the model is estimated from a certain population, a new population is sampled according this probabilistic model in order to continue with the evolutionary process. A Naïve Bayes model can be viewed

Algorithm 22 Naïve Bayes models Estimation (NBE) algorithm [LD05]

Input: Training set T, hold-out set H, initial number of components k_0, and convergence thresholds δ_{Add} and δ_{EM}

Output: Naïve Bayes Model M_{best} estimated from the input training set.

```
Initialize $M$ with one component.
$k \Leftarrow K_0$
repeat
  Add $k$ new mixture components to $M$, initialized using $k$ random examples from $T$
  Remove the $k$ initialization examples from $T$.
  repeat
    E-step: Fractionally assign examples in $T$ to mixture components, using $M$.
    M-step: Compute maximum likelihood parameters for $M$, using the filled-in data.
    if $\log Pr \sim M(H)$ is best so far then
      $M_{best} \Leftarrow M$
    end if
    Every 5 cycles, prune low-weight components of $M$.
  until $\log Pr \sim M(H)$ fails to improve by ratio $\delta_{EM}$
  $M \Leftarrow M_{best}$
  Prune low weight components of $M$.
  $k \Leftarrow 2 \times k$
until $\log Pr \sim M(H)$ fails to improve by ratio $\delta_{Add}$
Execute E-step and M-step twice more on $M_{best}$, using examples from both $H$ and $T$.
return $M_{best}$.
```

as a set of $|C|$ promising regions in the solution space, where probabilities $Pr(C = c)$ are an estimator of the size of each region and conditional distributions $Pr(X_i|C)$ a description of the individuals of these regions.

Given a model M represented by marginal probabilities $Pr(C)$ and conditional distributions $Pr(X_i|C)$, generating a new population of L individuals is performed by generating $|C|$ sub populations of $L_c|\forall c \in C\}$ individuals using the related conditional distributions. Since the conditional distributions assume independence between the random variables, the problem is reduced to sample a multinomial distribution for discrete variables and Gaussian distributions for continuous ones.

The number of individuals L_c generated according each conditional distribution, can be determined using two different criteria:

Cluster Probability: Since cluster probabilities $Pr(C = c)$ are proportional to the individuals used in the estimation of $Pr(X_i|C = c)$, we can assume that it exist a correlation between $Pr(C = c)$ and the quality of the individuals represented in $Pr(X_i|C = c)$. Therefore, the number of individuals is weighted with the cluster probability: $L_c = L \times Pr(C = c)$

Individuals value: Once the model is estimated from the group of selected individuals Ω, each individual $I \in \Omega$ is classified into their maximum likelihood cluster using the conditional distribution: $\text{argmax}_{c \in C} P(X = I, C = c)$. The result is a set of disjoint sub populations $\Omega = \bigcup_{c \in C} \Omega_c$. The mean fitness value μ_c for each sub population Ω_c is then calculated and the number of individuals to be generated from each cluster is set proportionally to the sum of all clusters mean: $L_c = \frac{\mu_c}{\sum_{c' \in C} \mu'_c} \times Pr(C = c)$.

First approach only takes into account the number of individuals represented by a certain cluster, that is, if there is a large group of similar individuals in the selected population, these individuals will have a large weight on the next population, independently of their fitness value. In contrast, second approach takes into account the fitness value, but can fail if small differences exist between best values and worse values, because the cluster containing both will have a low weight in the next population. Note that it is possible to combine both strategies in a hybrid sampling method.

4.8.4 Results

In order to test the new Probabilistic Darwin Machine, we developed a framework in C++, which has been published in Google Code[1] under GNU licence where we embedded the implementation for the NBE algorithm of Lowd & Domingos [LD05][2]. We first test this method to solve the standard problems defined in Section 3.4, where this method obtains the second best ranking, just after the PBIL algorithm. As in the previous case, the statistical study is not conclusive. The standard problems are a good test because it exist the best value and it is known, but have the limitation of being a binary problem with no relationship with the object detection approach. In the following, a small experiment with a synthetic object detection problem is performed in order to compare PBIL and PDMNBE algorithms in a more realistic manner. After these preliminary results, we test the new Probabilistic Darwin Machine for the object detection problem, following the same experimental setting used before.

Looking for the best needle in a haystack

Tests of algorithms in a complete object detection framework give us an idea of how good are these methods to learn a detection system. Nevertheless, there is still an open question regarding to the quality of given solutions: *How good is the weak hypothesis we get?*

To answer that question we prepare a synthetic detection problem based on the ARFace database [MB98], a database similar to the used in this thesis (see Section C.1), where in spite of having a lower number of images, they are bigger and contain gender information. We consider the problem of classifying each positive sample by gender, assuming that there is not a unique feature which performs this task with zero error.

[1] `http://code.google.com/p/eapmlib/`

[2] `http://www.cs.washington.edu/ai/nbe/`

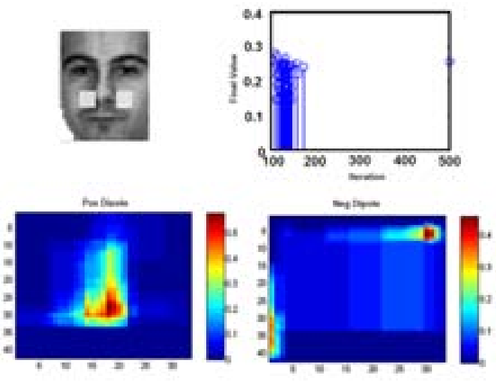
We create two disconnected homogeneous regions at each sample (R_1 and R_2), with values (V_1 and V_2) near the mean value of these regions in the image. These values are adjusted in order to obtain $V_1 - V_2 > 0$ when image corresponds to a man and $V_1 - V_2 < 0$ when image corresponds to a woman. Once these regions are displayed on the samples, any feature which compares these regions will classify all the samples, obtaining zero error. The size of these regions correspond to the difficulty parameter of the problem.

The experiment consists of finding the best instance of dissociated dipoles for this problem using PBIL and PDMNBE algorithms. In this case, all the samples are weighted with an equal value. Both algorithms are configured to use a population size of 200 individuals and a maximum of 500 iterations. Two experiments are performed, one using squared regions with size 5 and other using squared regions with size 12. Each experiment has been repeated 50 times, storing the error value at the stopping iteration and the returned best features. These results are shown in Fig. 4.18. Note that in general, PBIL converges to local minimum solutions, while PDMNBE is able to continue evolving. For the window size of 5 pixes none of the algorithms is able to find the good solution, while in the case of regions of 12 pixels, with PBIL we find the perfect solution the $33/50 = 66\%$ of the runs and with PDMNBE the $42/50 = 84\%$ of the runs. Moreover, if we look at the iterations plot, we observe that PBIL has converged in the most of the runs previous to the last iteration, while PDMNBE was stopped before convergence. Removing the iterations restriction, PBIL algorithms converged to the best value in $35/50 = 70\%$ of the runs, and PDMNBE in $50/50 = 100\%$ of the runs. In the accumulation graphics, we can see that in general PDMNBE obtains more defined peaks in the region areas than PBIL.

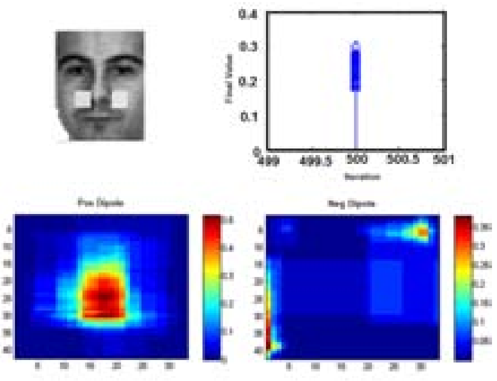
Object detection

In order to compare the Probabilistic Darwin Machines based on Naïve Bayes models Estimation (PDMNBE) with previous approaches, the experiments performed with Genetic Algorithms and Extended Compact Genetic Algorithms are repeated changing only the evolutionary algorithm. To allow comparisons, we maintain both the number of iteration for the AdaBoost algorithm at 200 and the number of individuals at 100. The definitions for problem parameters are the ones defined in Equations 4.1, 4.8, and 4.12, with their related constrains defined in Equations 4.2, 4.9, and 4.13. Moreover, the evaluation function defined in Algorithm 21 is used in these experiments, using a vector of random variables to encode the problem. The results for all the datasets are presented in Table 4.10.

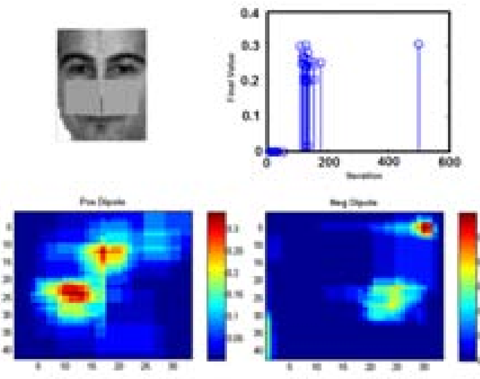
Notice that the mean rank of each feature remains the same that in the case of Genetic Algorithms and Extended Compact Genetic Algorithms. As we state in previous results, the representation power of each feature set is not altered with the use of different learning algorithms. In order to compare the performance of PDMNBE with GA and EcGA, all the results are summarized in Table 4.11, showing the mean rank for each feature set and the overall mean rank. As in previous sections, each combination of a dataset and a feature set is considered as an experiment, while the learning algorithms are taken as methods. The χ^2 and F_F statistics, and their related critical values are shown in Table 4.12.



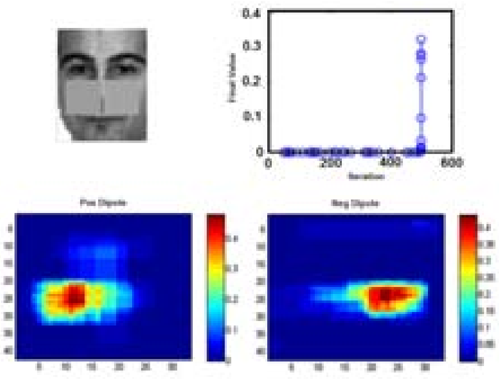
(a) PBIL (5 pixels)



(b) PDMNBE (5 pixels)



(c) PBIL (12 pixels)



(d) PDMNBE (12 pixels)

Figure 4.18: Results obtained by PBIL and PDMNBE in a synthetic gender classification problem. On the top the results using squared regions of 5 pixels and at the bottom using squared regions of 12. For each sub-figure, we show the error value at the stopping iteration and a mean image of the position of each pole of the final dissociated dipole at each run.

From those statistical values, we can state that the null-hypothesis can be rejected in the case of Haar-like features, Weighted Dissociated Dipoles and in the overall case. If we proceed with the Nemenyi post-hoc test, the critical difference in the case of Haar-like features is:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} = 1.48 \tag{4.20}$$

Since the rank differences between GA and EcGA is $2.8 - 2 = 0.8 < 1.48$, both methods are statistical equivalent. The rank difference between GA and PDMNBE is $2.8 - 1.2 = 1.6 > 1.48$, therefore we can state that for Haar-like features, PDMNBE is statistically significant better than GA. Finally, since the rank difference between EcGA and PDMNBE is $2 - 1.2 = 0.8 < 1.48$, there is no statistically significant

Table 4.10

Area under the ROC curve for the Evolutionary AdaBoost using Haar-like features, Dissociated Dipoles, and Weighted Dissociated Dipoles. Learning process is performed by an evolutionary AdaBoost with Probabilistic Darwin Machine based on Naïve Bayes models Estimation (PDMNBE) as weak learner.

	AUC		
	Haar-like	***Dissociated Dipoles***	***Weighted Diss. Dip.***
Cars	70.62% ± 4.98(3)	78.59% ± 7.50(2)	94.64% ± 4.81(1)
Faces	55.43% ± 4.93(3)	69.40% ± 5.09(2)	88.27% ± 5.25(1)
Text	47.92% ± 4.85(3)	50.48% ± 5.02(2)	83.27% ± 3.36(1)
Pedestrians	55.91% ± 8.24(3)	70.37% ± 3.72(2)	90.01% ± 1.89(1)
Traffic Signs	65.37% ± 3.92(2)	63.06% ± 3.91(3)	89.82% ± 2.51(1)
Mean Rank	2.8	2.2	1.0

Table 4.11

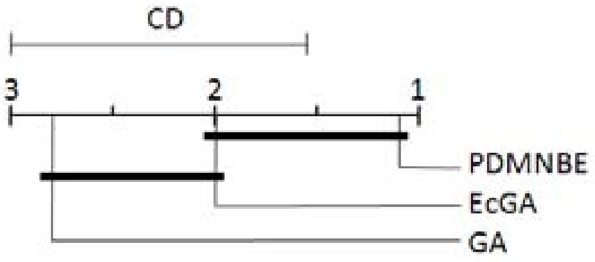
Area under the ROC curve for the Evolutionary AdaBoost using Haar-like features, Dissociated Dipoles, and Weighted Dissociated Dipoles. Comparative between Genetic Algorithms (GA), Extended compact Genetic Algorithms (EcGA), and Probabilistic Darwin Machine based on Naïve Bayes models Estimation (PDMNBE) as weak learners.

Experiment		**Method**		
Features	**Dataset**	**GA**	**EcGA**	**PDMNBE**
Haar-like	Cars	69.65%(3)	73.80%(1)	70.62%(2)
	Faces	55.22%(2)	52.96%(3)	55.43%(1)
	Text	45.84%(3)	47.83%(2)	47.92%(1)
	Pedestrians	54.00%(3)	55.39%(2)	55.91%(1)
	Traffic Signs	63.30%(3)	64.60%(2)	65.37%(1)
Mean Rank		2.8	2	1.2
Dis. Dip	Cars	79.27%(1)	76.19%(3)	78.59%(2)
	Faces	67.47%(3)	68.63%(2)	69.40%(1)
	Text	50.33%(3)	50.80%(1)	50.48%(2)
	Pedestrians	68.16%(3)	68.22%(2)	70.37%(1)
	Traffic Signs	61.53%(3)	63.13%(1)	63.06%(2)
Mean Rank		2.6	1.8	1.6
W.D.D.	Cars	95.21%(1)	85.99%(3)	94.64%(2)
	Faces	87.74%(2)	79.73%(3)	88.27%(1)
	Text	80.35%(2)	79.88%(3)	83.27%(1)
	Pedestrians	88.40%(2)	80.55%(3)	90.01%(1)
	Traffic Signs	87.92%(3)	89.71%(2)	89.82%(1)
Mean Rank		2	2.8	1.2
Mean Rank		2.4667	2.2	1.3333

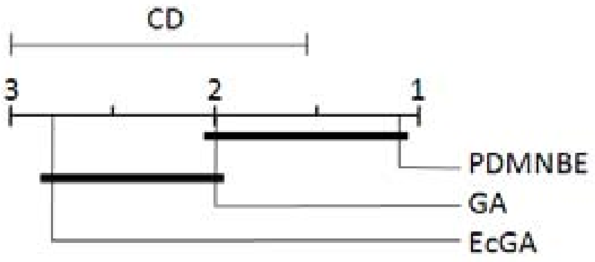
Table 4.12
Statistics and critical values for experiments in Table 4.11.

			CV $\rho = 0.05$		**CV $\rho = 0.1$**	
Features	$\chi^2_\mathbf{F}$	$\mathbf{F_F}$	$\chi^2_\mathbf{F}$	$\mathbf{F_F}$	$\chi^2_\mathbf{F}$	$\mathbf{F_F}$
Haar-like	6.4	7.11	5.99	4.46	4.61	3.11
Dis. Dip	2.8	1.56	5.99	4.46	4.61	3.11
W.D.D.	6.4	7.11	5.99	4.46	4.61	3.11
All	10.53	7.58	5.99	3.34	4.61	2.5

difference between these two methods. For the Weighted Dissociated Dipoles, we obtain the same critical difference $CD = 1.48$, and following the same process, we can state than GA and EcGA are statistically equivalents, and in this case, PDMNBE is better than EcGA. A graphical representation of this comparison is shown in Fig. 4.19, where the mean rank are plotted with the critical difference. Finally, the critical



(a) Haar-like features



(b) Weighted Dissociated Dipoles

Figure 4.19: Comparison of different evolutionary algorithms as weak learners in the AdaBoost framework, using the Nemenyi test. Groups of methods that are not significantly different (at $\rho = 0.05$) are connected.

difference in the global comparison is $CD = q_\alpha\sqrt{\frac{k(k+1)}{6N}} = 0.86$. If we analyze the overall rank differences, we can state that PDMNBE performs statistically better than GA ($2.47 - 1.2 = 1.27 > 0.86$) and EcGA ($2.2 - 1.2 = 1.0 > 0.86$), while non difference exist between GA and EcGA ($2.47 - 2.2 = 0.27 < 0.86$). The rank differences are represented together with the critical difference in Fig. 4.20.

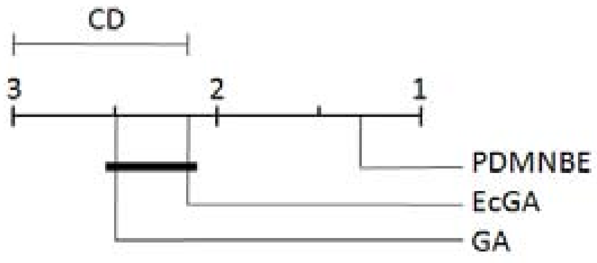


Figure 4.20: Comparison of different evolutionary algorithms as weak learners in the AdaBoost framework, using the Nemenyi test. Groups of methods that are not significantly different (at $\rho = 0.05$) are connected.

4.9 Learning a cascade of detectors

Once the usefulness of evolutionary computation in the Adaboost framework has been demonstrated, the next step is to learn an attentional cascade of evolved detectors. As explained in Section 2.7.5, an attentional cascade is an organization architecture where each stage is a detector which are trained to concentrate on those samples that have been misclassified by previous stages.

The goal of this final experiment is to find out the benefits of evolutionary computation on the whole classical approach. For this proposal, we train a cascade using a weak learner based on exhaustive search with Haar-like features, and a cascade using an Evolutionary weak learner with Weighted dissociated dipoles. In both cases, each stage is trained in order to obtain a *minimum hit ration* of 99.5% and a *maximum false alarm* of 40%. Since the canonical problem for object detection is the face detection problem, both cascades are trained using all the positive samples of the face detection dataset (see Section C.1), and negative samples are generated each stage from the Corel Image Database (see Section C.6.

The theoretical hit ratio for both cascades must be higher than $HR = S_{HR}^n = 0.995^5 = 97.5\%$ and a false alarm lower than $FA = S_{FA}^n = 0.4^5 = 1.02\%$, where n is the number of stages in the cascade and S_{HR} and S_{FA} are the hit ratio and false alarm for each stage respectively. Although these values are not enough for a detection system, where we need at least a false negative ration below $10^{-3}\%$, this experiment allows to compare how the learning process is evolving. In order to improve these values we just need to add more stages to the cascades.

In the case of the Evolutionary weak learner, we used the Probability Darwin Machine based on Naïve Bayes Models to select weak hypotheses. For this algorithm, we set a population size of 200 individuals and a maximum number of iterations of 300. In Table 4.13 the number of weak hypotheses used for each algorithm in order to achieve the hit ration and false alarm of each stage are presented. Notice that in general the evolutionary algorithm need a lower number of weak hypotheses for the same parameters. By mean, the evolutionary algorithm reduce the number of hypothesis in a 17.3% respect to the classical approach.

Table 4.13
Number of weak hypothesis for each stage of the cascade. Last row shows the variation in the number of weak hypothesis in the case of the Evolutionary weak learner with Weighted dissociated dipoles respect to the Exhaustive weak learner with Haar-like features.

	Stages				
Experiment	**1**	**2**	**3**	**4**	**5**
Haar-like	8	27	53	74	120
W.D.D.	5	22	56	68	100
Variation	$\Downarrow$ 37.5%	$\Downarrow$ 18.5%	$\Uparrow$ 5.6%	$\Downarrow$ 18.9%	$\Downarrow$ 16.6%
♯Eval. Haar	$1200K$	$4050K$	$7950K$	$11100K$	$18000K$
♯Eval. WDD	$300K$	$1320K$	$3360K$	$4080K$	$6000K$
Variation	$\Downarrow$ 75%	$\Downarrow$ 67.41%	$\Downarrow$ $57,74$%	$\Downarrow$ 63.24%	$\Downarrow$ 66.67%

Apart from the number of weak hypothesis, there are two other important indicators to compare both approaches: The real final performance and the training time. In the case of final performance, both approaches obtains similar values, slightly better than the theoretical values. Moreover, taking into account that images have a size of 20×20 pixels, there are more than 150.000 valid Haar-like features that need to be evaluated in order to find the best weak hypothesis at each iteration of the Adaboost. In the case of PDMNBE, in the worse case, we need to evaluate the 200 individuals 300 times before to return the final weak hypothesis. Using these numbers, in Table 4.13 we show the number of evaluations performed by each algorithm to learn the whole stage and the variation. The large amount of extra computation in the exhaustive algorithm is compensated in part using precalculated features. Because of that, these large differences on the calculation effort are not reflected on the training time.

Respect to the training time, the exhaustive algorithm spends five days to learn the cascade, while the evolutionary one spends only four days. That small difference is because the most consuming task is to find the negative samples for each stage, that is, using a proportion of 1 : 3 between positive samples and negative samples in the training set, and using 2500 positive samples, we need to find 7500 false positive samples for each stages. Since the false positive ratio becomes slower stage by stage, each time is more difficult to find them. For the last stage, we analyzed more than 300.000 images in order to find 7500 images where the first 4 stages fail. Therefore, this time difference will become more insignificant as the number of stages increases.

Finally, in Fig. 4.21 some negative samples discarded at each stage are shown. Notice that images becomes more complex stage after stage. Images at the bottom of the figure are false positive, which should be used on the training of the following stage.

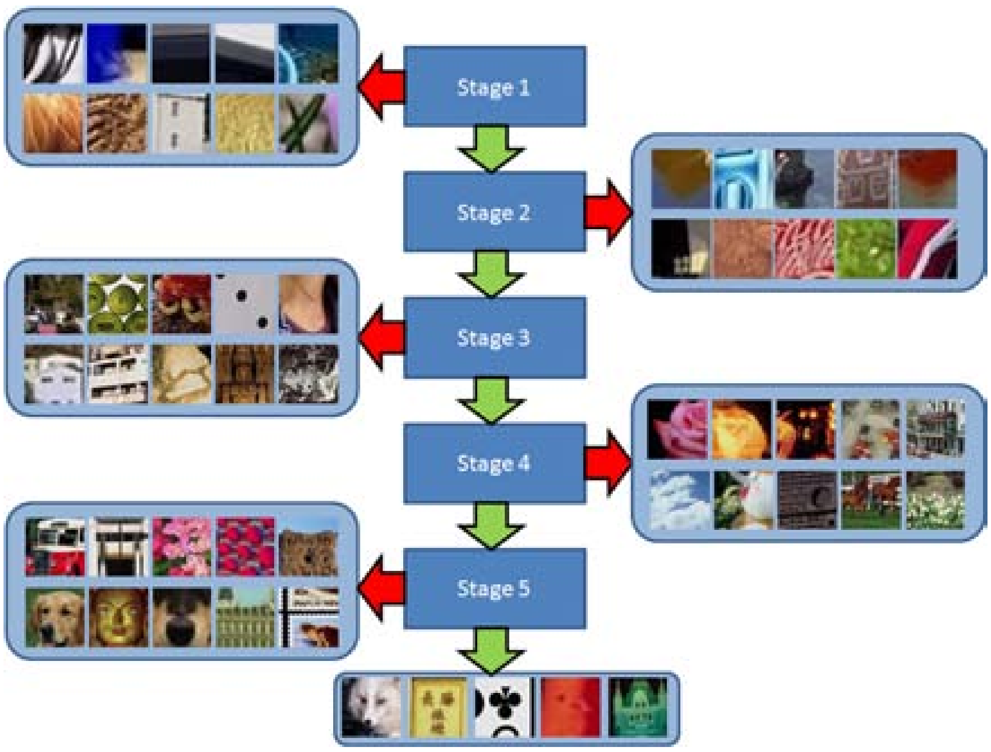


Figure 4.21: Images discarded at each stage of a face detection cascade.

Chapter 5

Conclusions and Future Work

After a wide review of the state-of-the-art literature for object detection and evolutionary computation, a new framework which combines methodologies from both fields has been proposed. In this section, the conclusions emerged from this new framework are highlighted. Since the proposed framework shares methodologies from two different fields, it is possible to continue this research in both fields, and we propose some possible future lines from both points of view: Object Detection and Evolutionary Computation.

5.1 Conclusions

The most important conclusion which arises from obtained results is that the classical approach to object detection proposed by Viola & Jones [VJ01] and commonly applied to a wide diversity of object detection problems can be improved by using larger feature sets. In this thesis, a general framework for object detection has been defined based on evolutionary computation, which has been demonstrated to be reliable for different problems. Moreover, this framework opens different improvement lines based on the definition of new features which were to complex for the classical approach and using new evolutionary algorithms which obtain better weak classifiers.

From the object description point of view, the obtained results show that the robustness of AdaBoost methodology allows to learn a good detector even if weak hypothesis are not the best available ones, which is reflected in the fact that the most important performance improvements are related to how the objects are described, not in the weak learner algorithm. Nevertheless, the test of the new proposed evolutionary algorithm over the standard problems demonstrated a good adaptation of this method to different situations.

5.2 Contributions

The contributions in this thesis are basically tree:

Framework: A new evolutionary framework for object detection has been presented, which breaks the limitation in the classical approach of using large cardinality feature sets. Although we state that a large cardinality is not a synonym of better results, it allows to define feature sets which adapt to the goal problem with no limitation on their cardinality.

Features: We defined the Weighted Dissociated Dipoles, a new feature set which shares the benefits of two classical feature sets, obtaining a better description power and thus better results.

Evolutionary Algorithm: A new evolutionary algorithm has been proposed, which demonstrated to better learn the proposed evolutionary object detection framework. Although the underlying idea of using Naïve Bayes Models in Probabilistic Darwin Machines is not new, and is used in the EBCOA framework (see Section 3.3.5), there are two main differences with EBCOAs: How model is estimated and how the scores are used in the generation process.

In the EBCOAs framework Naïve Bayes models are used to classify individuals according their scores in a predefined number of clusters, learning a model for each one of the clusters, which introduce a hard dependency on a parameter which is strongly related to the selected individuals. In PDMNBE the introduction of one state-of-the-art algorithm defined by the statistics community, allows an automatic estimation of the clusters, allowing to better represent the individuals independently of their score. Moreover, the key idea in EBCOAs is to use the score not only on the selection process, but in the generation of new individuals. This idea is maintained in PDMNBE when the score-based sampling method is used on the generation of new individuals.

Although in the experiments comparing the new evolutionary algorithm to the classical approach, we show that most part of the learning time is consumed looking for false negative samples, the use of an evolutionary algorithm reduces the time of learning the classifiers, independently of the feature set cardinality.

5.3 Future Research

The work presented in this thesis opens the door to other research lines that we would like to explore in a future. In addition to the slight improvement on the results obtained using PDMs instead of classical Genetic Algorithms, it is a first step to other research lines to speed-up the learning process and the addition of problem dependent information. This chapter introduce the basic ideas about that future steps.

5.3.1 Evolutionary Learning

The first two new lines explore the possibilities of the use of any evolutionary approach in the detector learning process.

New features

As we show with the dissociated dipoles and weighted dissociated dipoles, this new approach is able to effectively learn detectors based on huge cardinality feature sets. In fact, our work suggest that any feature set which can be parameterized can be used to build a detector. It means that new features that cannot be used in object detection due to the learning complexity can be tested, obtaining more powerful descriptors for the objects.

Multi-class approach

The problem of multi-class object detectors usually is addressed using multiple specific detectors. This is because in general the multi-class versions of AdaBoost get poorer results than binary ones. In the case of classification, one of the most effective approaches to address the multi-class problem is to combine multiple binary classifiers using an ECOC strategy.

Recently, Torralba et al. [TMF07] presented a multi-class Boosting strategy based on feature sharing among classes, the Joint boosting. The idea is to create all possible partitions of classes, and each new *week classifier* votes to one of those partitions, similar to the ECOC strategy. Although the object description in this work is based on a *bag of words*, this idea can be used with any family of features.

As in the case of the classical AdaBoost, this method is limited by the feature set cardinality, and in addition, the number of possible partitions increase exponentially with the number of classes. Using the evolutionary strategy, we can codify the sharing information as part of the feature parameters, removing the limitations of a greedy search approach.

5.3.2 Learning based on Probability Models

When the evolutionary AdaBoost is based on EAPMs, the probabilistic models are only used in order to evolve the features, and the only result of this process are the final features. The following lines are focused to take advantage of those probability models used during the learning process.

A priory models

Although most probabilistic models can be used to learn the features, the Bayesian based models have the advantage to be guided using prior information. In the work of Gallagher et al. [GWKS07], an EAPM based on Bayes Inference is presented, using priors in order to guide the learning process. Although this paper only presents a simple example using an univariate model, and a non informative prior, the use of priors in the EAPM framework can be extended to more complex probability models. From our point of view, the use of priors can be interesting in two ways:

- Add prior knowledge about the problem. In many problems, especially when our data are images, it is possible to extract information about the problem. For instance, if we calculate the variance over all the positive samples, the regions with large variances probably contains no interesting information of the target objects, and this information can be used in the process of generating a feature.

- In the EAPM learning process, the models are discarded at each iteration, inferring a new model from the selected samples. It can be interesting to use information of the last models as prior for the new estimated model, smoothing the learning process.

In both approaches, the most important issue to resolve is how that knowledge is represented and used in the inference process.

Looking for the forest behind the trees

There are different aspects in the Probability Darwin Machines that should be deeply studied, such as all these related to the stability of evolutionary algorithms and how different probability models can be compared and evaluated. In other words, find methodologies to see the population and not the individuals. As in many situations, we need to deal with the *Rashomon effect* [Bre01] related to the subjectivity of perception. During evolution process, we find different models to represent the promising areas of the solution space, some of them could be complementary, but others may be contradictory. When multiple sources relate different and sometimes conflicting accounts of the same problem, how do we decide which one is the "right" one? Is it possible that they all are right? In other words, can we obtain a better explanation of our problems using the points of view of different models?

One of the points where this effect becomes evident is during the learning of a certain classifier using the Adaboost algorithm. On a certain moment we need to add a new weak hypothesis in order to better explain some problem. It is plausible that we can find for instance two weak hypothesis with the same error value but which produces contradictory hypotheses for the same samples. In this situation, which is the best decision? In the case of Adaboost, this situation is not considered, choosing one of these hypotheses in arbitrary manner. A possible solution is to look the forest instead of the trees, looking for the final ensemble of weak hypotheses instead of just the next one. In the same way, we can think on learning the whole cascade instead of their individual stages, and so on. The first step needs to codify not only a fixed set of parameters which represents a weak hypothesis, but a set with an unknown number of weak hypotheses and how they are related. This problem can be seen as atomic particles (the weak hypotheses) which are combined using some type of operator (sum in the case of Adaboost), and this is a typical problem solved with Genetic Programming.

On-line learning

During the learning process, a probability model is estimated for each final feature. Those probability models in our opinion represent the promising regions of the solution space for a certain feature. Therefore, at the end of the learning process, we have an ensemble of probability models, which can be used in order to sample new features without to repeat the whole learning process, sampling the models of the features that do not match the new samples looking for another promising point in the feature space that adapts better those new samples.

Appendix A

Traffic Sign Detection

In this appendix we describe a real application where some of the object detection methods introduced in this thesis have been applied together with the results obtained by Sergio Escalera in his thesis [Esc08], more related to the mutli-class classification strategies. Traffic sign recognition is studied for several purposes, such as autonomous driving or assisted driving [HKT$^+$98, FGG$^+$98]. Recognition of traffic signs allows warning the driver for inappropriate actions and potentially dangerous situations. In the mobile mapping framework, traffic sign recognition methods are used in combination with other methods in order to compile road information and measuring position and orientation of different landmarks in movement either in an aerial or a terrestrial platform. An example of this system is given by Madeira et al. [MBS$^+$05], where a mobile mapping system automatically processes traffic signs. In this work, a recognition accuracy over 80% on a reduced set of sign types is obtained. In [BZ04], a vehicle based vision platform is used to detect road signs, where the main goal is mainly focused on speed signs.

In the literature, we can find two main approaches to solve the problem of road sign recognition: color-based and grey scale-based sign recognition. The first one relies on color to reduce false positive results in the recognition process [AI87, KZ94, dSB92, EMSA97, GLY94, dlEAS01, MKS00, SPG$^+$02, HH01], whereas the greyscale methods concentrate on the geometry of the object [PMC96, PMPC94, ER04, LZ03]. Recent works use combination of both cues to improve the detection rates. For instance, in [MBLAGJ$^+$07] a threshold is applied over a HSV representation of the image to find regions with high probability of having a traffic sign. As many background objects can share colors with traffic signs, heuristics over the size and aspect ratio are used to reduce the number of false alarm regions. Once the regions are normalized to a predefined size, a linear SVM is used to classify the region in one of the possible shapes, such as circle or triangle. The color and shape information are used as a coarse classification, and finally a SVM with Gaussian kernels is used to perform the fine classification step. Since the color information is strongly related to the type of camera, illumination and sign aging, the use of color information introduces additional difficulties to the recognition process. In the work of A. de la Escalera et al. [dlEAPR04], these difficulties are addressed using an enhancement step previous to the use of thresholds on the color values. After applying size heuristics to remove non-sign regions, the authors use a fusion of color information, the gradient, and a distance image to remove regions with low probability of having a traffic sign. Final classification is performed by means of a Neural Network. Other recent works are focused on the final classification step. The authors of [PND06] propose a representation of road sign data based on extending the traditional normalized cross correlation approach to a similarity based on individual matches in a set of local image regions.

Traffic sign recognition is a straightforward application for object recognition algorithms in which previous addressing of the category detection (e.g. object location) is often required. In the last years, one of the most accepted and used approaches in the object detection field has been the one proposed by Viola & Jones in [VJ01]. Their approach is based on a cascade of detectors, where each one is an ensemble of boosted classifiers based on the Haar-like features (see Section 2.7 for deeper analysis of this approach). Lienhart and Maydt [LM02] presented an extension of the original Haar-like features set, demonstrating that *Adaboost* converges faster and with better results when the features set is large. On the other hand, due to the exhaustive search over the features set, the training time grows with respect to the number of features. This fact makes unfeasible any approach that tries to extend the feature set.

Once an object (traffic sign) is located, it should be recognized from a wide set of possible classes using some kind of classification technique. Designing a machine learning multi-class technique is a hard task. In this sense, it is common to conceive algorithms to distinguish between just two classes and combine them in some way. Following the multi-class categorization problem, where a set of classifiers should learn in a natural way the features shared between categories, the Error Correcting Output Codes technique was proposed with very interesting results [DB05]. This technique is a very successful multi-class categorization tool due to its ability to share the classifiers knowledge among classes. Recently, the embedding of a tree structure in the ECOC framework has shown to obtain high accuracy with a very small number of binary classifiers [PRV06]. However, the ECOC design is still an open issue. Since the goal of this thesis concerns to the detection, the final classification of detected traffic signs is out of scope. Further information about the classification strategies can be found in [Esc08].

In the following, we perform an introduction to the mobile mapping system used in this project, and finally the methodology and results obtained for the traffic sign detection step.

A.1 Mobile Mapping System

The mobile mapping system used in the project belongs to the ICC[1], which is developing its own mobile mapping system, named Geomobil [ABB$^+$04]. This system incorporates inside a van all the sensors required for the capture of stereo-pairs of digital images and their subsequent georeferencing for the extraction of information (Fig. A.1). The Geomobil includes an image-capture subsystem based on a pair of digital cameras of 1020×1024 pixels, a direct image orientation subsystem based

[1]ICC (*Institut Cartogràfic de Catalunya*) www.icc.cat

on GPS/INS[2] and a synchronization subsystem. The cameras are calibrated to determine the GPS/INS orientation misalignment and to correct the errors due to the distortion of the optics of the digital cameras.

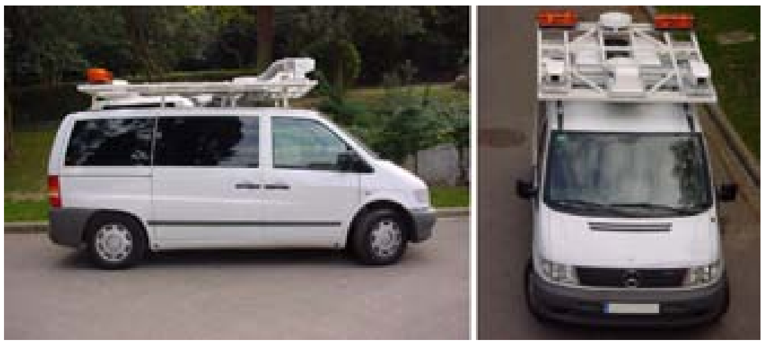


Figure A.1: Geomobil system.

A.2 Problem analysis

First step on the project consists of analyze the images provided by Geomobil. Looking to the captured images (see Fig. A.2), one can observe that images suffer from hard illumination changes due to road artifacts and weather conditions. Lighting corrections over the whole image were discarded because the final result is degraded when images suffer from local perturbations, as large shadows. In addition, road sequences contains a large variety of objects as cars, trucks, advertisements, buildings near road, etc... which often contains structures with shapes similar to traffic signs.

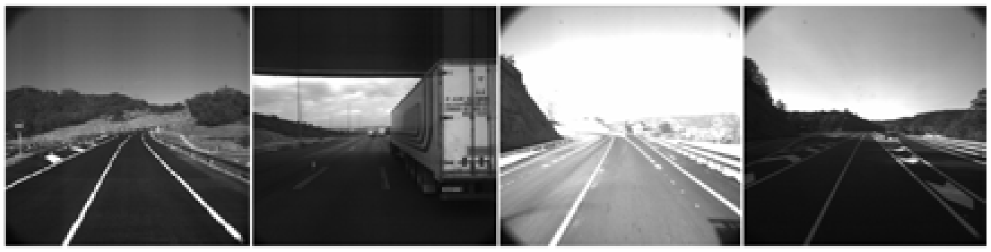
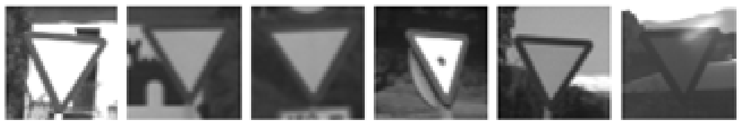


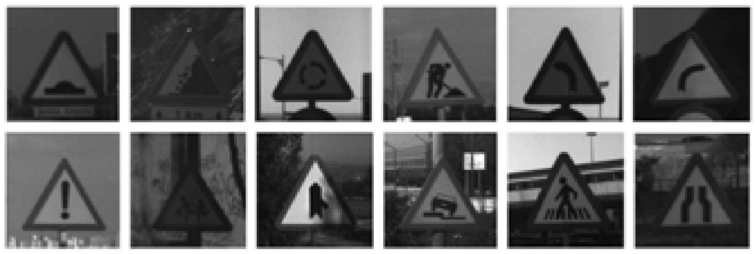
Figure A.2: Some examples of images acquired by Geomobil system. From left to right, we see a normal illuminated image, effect of road artifacts (bridge), light image and dark image.

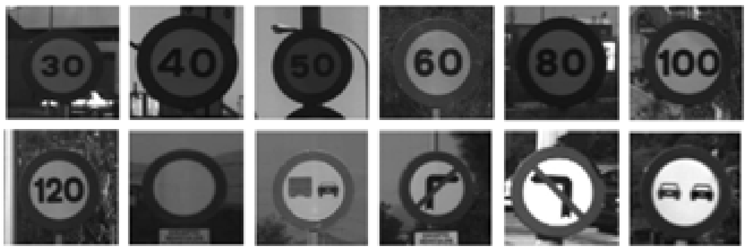
Moreover, after analyzing the instances of objects in the images, we detect a large inter-class and intra-class variability (see Fig. A.3), with large differences on their apparition frequencies. From a practical point of view, the differences on the objects

[2]Global Positioning System / Inertial Navigation Systems

aspect requires flexible methods which allows quite different images being considered as the same object, and at the same time, enough discriminative in order to filter out the artifacts contained on the background. Respect to the different apparition frequencies, the main problem is that some traffic sign appears in any road, as speed signs, while other signs only appears on specific situations, as in tunnels, mountain roads or in roads under construction. Therefore, if the sequences used as training do not contain some of those special signs, the learned system could have problems to recognize it.


(a) Yield signs.


(b) Danger signs.



(c) Prohibition signs.

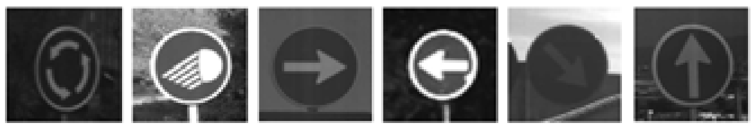

(d) Command signs.

Figure A.3: Instances of considered traffic sign classes.

A.3 Detection approach

After the considerations extracted from the problem analysis, the evolutionary object detection approach presented in Section 4 is adopted as the base of our system. This system is able to detect objects allowing aspect variations, but when the objects are too different, performance is reduced. To minimize the variability, signs are grouped by their similarity, and a cascade of detectors is learnt for each group of similar signs: prohibition, command, yield and danger signs (see Fig. A.14). From a recognition point of view, a part from the object detection process itself, the use of parallel and independent cascades, result in a first classification of the objects in one of the four groups. Other groups of signs are not considered due to the lack of examples, and can be added to the system by simply adding additional cascades.

A.4 Stereo Association

Since we work with a stereo system, all signs appear on both cameras at each frame (except in case of occlusions or if one of the signs is out of the field of view). This redundant information is used in order to improve the detection ratio.

Using the epipolar geometry, given an instance of a sign in one of the sources, we estimate the region where it must appear in the other source. Once we have a reduced search window in the other source, we apply similarity criterion based on normalized correlation. The point with the highest similarity value gives us the position of the target object. This information is used to link the object of a source with its stereo object to recover it. Using this information, we only loose the objects that have been lost in both cameras.

Using the calibration data, the position and orientation information of the GPS/INS system, and the coordinates of an object in both cameras, we compute the object position in world coordinates.

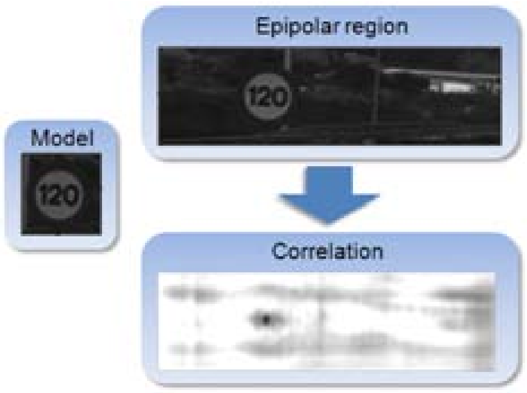


Figure A.4: Correlation between a sign detected in one camera (Model) and the corresponding epipolar region in the other camera.

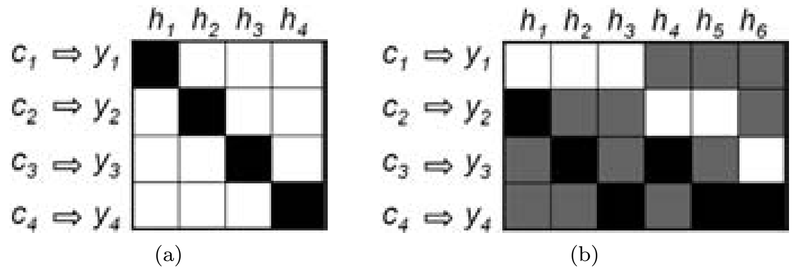


Figure A.5: Four-class ECOC designs. *(a)* One-versus-all ECOC codification and *(b)* one-versus-one ECOC codification (white: 1, black: -1, grey: 0).

A.4.1 Classification: Forest-ECOC

Once we located an object, we need to categorize among a large set of classes. Although various systems of multiple classifiers were proposed, most of them use similar constituent classifiers, which are often called base classifiers (dichotomies from now on). In this sense, Error Correcting Output Codes represent a classification technique that allows a successful combination of base classifiers to address the multi-class problem [DK95, DB05].

Error Correcting Output Codes

The design of an Error Correcting Output Code is based on a coding and a decoding strategy, where coding aims in assigning a codeword[3] to each of the N_c classes (up to N_c codewords), and decode aims in assigning a class label to a new test codeword. Arranging the codewords as rows of a matrix, we define the *coding matrix* M, where $M \in \{-1, 1\}^{N_c \times n}$, being n the code length. From the point of view of learning, the matrix M represents n binary learning problems (dichotomies), each corresponding to a column of the ECOC matrix M. Each dichotomy defines a sub-partition of classes, coded by $\{+1, -1\}$ according to their class membership. In Fig. A.5(a) the codification for a four-class problem using the one-versus-all coding strategy is shown. The white and black regions correspond to $+1$ and -1 valued positions, respectively. Thus, in (a), the dichotomy h_i is trained to discriminate class c_i against the rest of classes. If we use a larger set of symbols for coding $M \in \{-1, 0, 1\}^{N_c \times n}$, some entries in the matrix M can be zero, indicating that a particular class is not considered for a given dichotomy. In Fig. A.5(b), the codification for a four-class problem using one-versus-one coding strategy is shown. The grey regions correspond to the zero value (non-considered classes for the classifiers). In this strategy, all possible pairs of classes are split. For example, dichotomy h_1 classifies class c_1 versus class c_2, etc.

As a result of the outputs of the n binary classifiers, at the decoding step a code is obtained for each data point in the test set. This code is compared to the base codewords of each class defined in the coding matrix M, and the data point is assigned to the class with the "closest" codeword. The common distances to decode are the

[3]A codeword is a sequence of bits that represents a class.

Hamming and the Euclidean distances [ASS02].

Forest-ECOC

Most of the discrete coding strategies up to now are pre-designed problem-independent codewords (one-versus-all [Nil65], one-versus-one [HR98]). In the work of Pujol et al. [PRV06], a method for embedding tree structures in the ECOC framework is proposed. Beginning on the root containing all classes, the nodes associated to the best partition in terms of the mutual information are found, and the process is repeated until the sets with a single class are obtained.

Taking the previous work as a baseline, we propose to use multiple trees embedding, forming a Forest-ECOC. We build an optimal tree - the one with the highest classification score at each node - and several suboptimal trees - the ones closer to the optimal one under certain conditions. Let us keep at each iteration the best k partitions of the set of classes. If the best partition is used to construct the current ECOC tree, the rest of partitions form the roots of $k-1$ trees. We repeat iteratively this process until all nodes from the trees are decomposed into one class. Given a base classifier, the sub-optimal tree candidates are designed to have the maximum classification score at each node without repeating previous sub-partitions of classes. In the case of generating T first optimal trees, we can create an ensemble of trees by embedding them in the ECOC matrix, as shown in Algorithm 23.

Algorithm 23 Training algorithm for the Forest-ECOC.

```
Given $N_c$ classes: $c_1, ..., c_{N_c}$ and $T$ trees to be embedded
$\Omega_0 \Leftarrow \emptyset$
$i \Leftarrow 1$
for $t = 1, .., T$ do
  Initialize the tree root with the set $N_i = \{c_1, ..., c_{N_c}\}$
  Generate the best tree at iteration $t$:
  for each node $N_i$ do
    Train the best partition of its set of classes $\{P_1 P_2\} | N_i = P_1 \cup P_2, N_i \notin \Omega_{t-1}$
    using a classifier $h_i$ so that the training error is minimal
    According to the partition obtained at each node, codify each column of the
    matrix $M$ as:
    $$M(r,i) = \begin{cases} 0 & \text{if } c_r \notin N_i \\ +1 & \text{if } c_r \in P_1 \\ -1 & \text{if } c_r \in P_2 \end{cases}$$
    where $r$ is the index of the corresponding class $c_r$
    $\Omega_t \Leftarrow \Omega_{t-1} \cup N_i$
    $i \Leftarrow i + 1$
  end for
end for
```

The proposed technique provides a sub-optimal solution because of the combination of robust classifiers obtained from a greedy search using the classification score. One of the main advantages of the proposed technique is that the trees share their information among classes in the ECOC matrix M. It is done at the decoding step by considering all the coded positions of a class jointly instead of separately. It is easy to see that each tree structure of N_c classes introduces $N_c - 1$ classifiers, that is far from the $\frac{N_c \cdot (N_c - 1)}{2}$ dichotomies required for the one-versus-one coding strategy.

An example of two optimal-trees and the Forest-ECOC matrix for a toy problem is shown in Fig. A.6. The Fig. A.6(a) and (b) show two examples of optimal trees. The second optimal tree is constructed based on the following optimal sub-partitions of classes. In this way, for the first initial set of classes $\{c_1, c_2, c_3, c_4\}$, the two optimal trees include the best sub-partitions of classes in terms of the classification score, that in the example corresponds to c_1, c_3 vs c_2, c_4 for the first tree, and c_1, c_2, c_3 vs c_4 for the second tree, respectively. Fig. A.6(c) shows the embedding of trees into the Forest-ECOC matrix M. Note that the column h_3 corresponds to the node N_3, and the following dichotomies correspond to the nodes of the second tree. The classes that do not belong to the sub-partitions of classes are set to zero. On the other hand, the classes belonging to each partition are set to $+1$ and -1 values, defining the subset of classes involved on each classifier.

Recent studies on the decoding steps have shown that the zero symbol introduces decoding errors in the traditional decoding distances [EPR06]. To deal with this problem and to increase the performance of the Forest-ECOC coding design, we propose the Attenuated Euclidean decoding strategy, defined as $d_j = \sqrt{\sum_{i=1}^{n} |y_i^j|(x_i - y_i^j)^2}$, where d_j is the distance to row j, n is the number of dichotomies, x_i is the response of the classifier h_i over the test sample, and y_i^j is the value of the coding matrix M at i^{th} row and j^{th} column, respectively. We introduce the factor $|y_i^j|$ to avoid the error that the zero symbol introduces.

A.5 Results

To evaluate the detector performance, we train a cascade of detectors using the evolutionary method with ordinal dissociated dipoles. In Fig. A.7 we show the most relevant features selected by the evolutionary method at the first stage of the cascade. Note that only few of them correspond to Haar-like features.

Due to the different appearance frequency of each type of sign and the high intra-class variability, we trained a detection cascade for each group of similar signs. In table A.1 we show the groups of signs and the number of positive samples used to train each cascade. The number of negative samples on the train process is automatically selected at each stage with a proportion of $3:1$ (three negative examples for each positive example). Most part of the captured images are from main roads, and consequently, some types of signs do not appear enough times to train a detector. Due to this reason, we only trained the four detectors shown in table A.1.

The results are analyzed using two configurations. The first uses the stereo association to take advantage of the stereo information. The second considers each stereo-pair of images as two independent images. For each configuration, the obtained results with and without using sequential information are extracted. When the sequential information is used, different instances of the same real traffic sign are considered as the same object. In case of not using this information, each instance is

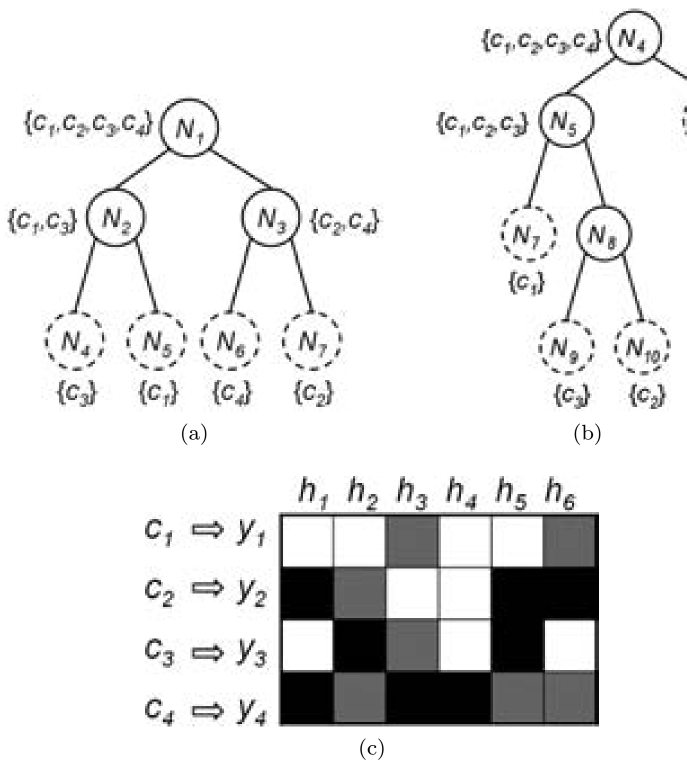


Figure A.6: Four-class optimal trees and the Forest-ECOC matrix. *(a)* First optimal tree for a four-class problem, *(b)* Second optimal tree for the same problem, and *(c)* Forest-ECOC matrix M for the problem, where h_1, h_2 and h_3 correspond to classifiers of N_1, N_2 and N_3 from the first tree, and h_4, h_5 and h_6 to N_4, N_5 and N_8 from the second tree.

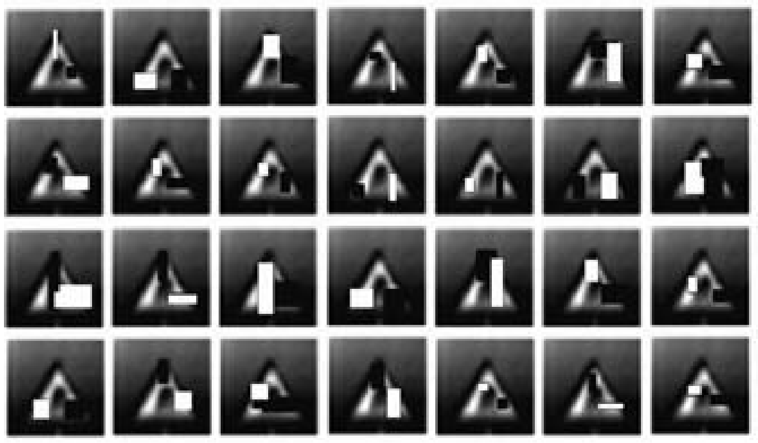


Figure A.7: Selected dipoles obtained over the danger signs.

Table A.1
Number of positive samples used to train the cascade for each considered sign.

Sign	Danger	Yield	Command	Prohibition
#Samples	545	425	256	993

considered as an independent object. In Fig. A.8, we show the hit ratio of the detector trained for each type of sign. In general, we can see that the accuracy of the detectors depends on the variability of sign appearance and the size of the training set. The First and the third columns correspond to the results considering each appearance of a traffic sign as a different sign. And the second and the fourth columns only take into account the real traffic signs, considering that a sign is detected if we can detect it in one or more frames where it appears. The first two columns do not take into account stereo redundancy, whereas the two last columns take it into account.

The other measure to evaluate the performance of the system is the false alarm rate. As we work with a mobile mapping system, an important point is which percentage of the detected objects corresponds to traffic signs. Therefore, our false alarm value is referred to the detected signs instead of the number of analyzed windows, which is of order of 5.000.000 per stereo-pair. Nevertheless, the number of false positives with respect to the number of stereo-pairs images has been included to make easier the analysis of the results. Both false alarm rates for each type of sign are detailed in table A.2. Some samples of detected objects and false alarms are shown in Fig. A.9. One can see that the system is able to detect the signs in a very extreme lighting conditions. In the false positive images, one can see that frequently, other road elements look similar to traffic signs.

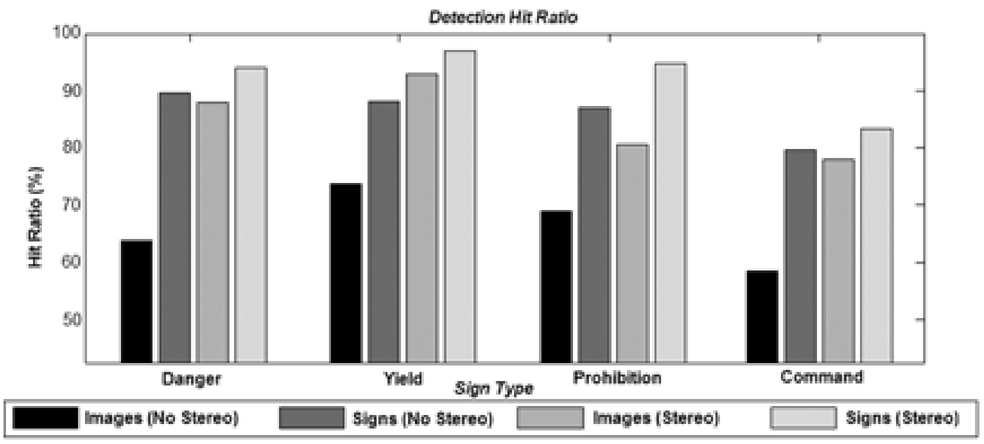


Figure A.8: Hit ratio for each sign type, using dissociated dipoles.

Table A.2
False alarm rates for each sign type.

Sign	Danger	Yield	Command	Prohibition
FA/Sign	2.140	4.549	8.551	0.696
FA/Frame	0, 045	0, 056	0, 073	0, 019

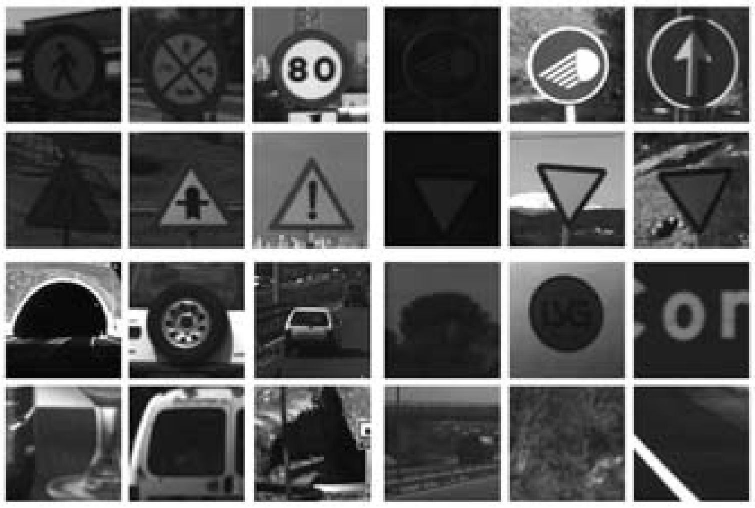
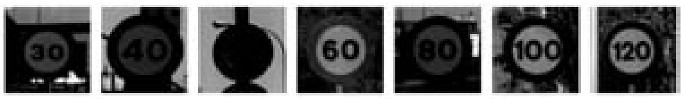


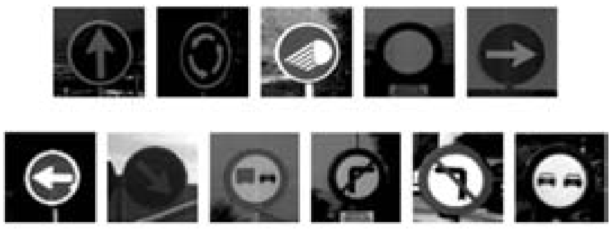
Figure A.9: Some samples of detected objects and false positives.

Classification Database

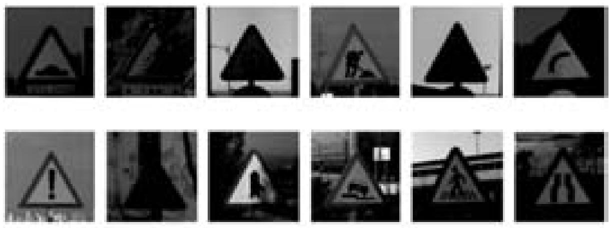
The database used to train the classifiers was designed using the regions of interest obtained from the detection step and the model fitting methods presented in the previous sections. We defined three groups of classes using the most common types of signs. The considered classes are shown in Fig. A.10. Speed signs need special attention. These types of signs are less discriminative, being some of them only differentiated by a few pixels. With this type of signs it is better to work on binary images to avoid the errors that can be accumulated because of the grey levels of the signs. For the twelve classes of circular signs and twelve of triangular signs we have 750 training images in both cases. For the seven speed classes we use 500 training samples. Finally, the resolution of each database is: 35 × 35 pixels for the circular group, 44×39 pixels for the triangular group, and 41×41 pixels for the speed group, respectively.



(a)



(b)



(c)

Figure A.10: Set of classes considered in the classification module. *(a)* Speed classes, *(b)* circular classes, and *(c)* triangular classes.

State-of-the-art comparison

To evaluate the Forest-ECOC performance, we compare it with the state-of-the-art classifiers. The details for each strategy are: 3-Euclidean distance Nearest neighbors (K-NN), Tangent Distance (TD) [SLDV98] with invariant tangent vector with respect to translation, rotation, and scaling, 99.98% of Principal Components Analysis followed by 3-Nearest neighbors (PCA K-NN) [DFS00], Fisher Linear Discriminant

Analysis with a previous 99.98% PCA (FLDA) [DFS00], Support Vector Machine with projection kernel Radial Basis Function and the parameter $\gamma = 1$ (SVM) [HCL02], Gentle Adaboost with decision stumps using the Haar-like features (BR) [LM02, KHZ00], multiclass Joint Boosting with decision stumps (JB) [TM04], Gentle Adaboost [FHT00] Sampling with FLDA (BS), statistical Gentle Naive Boosting with decision stumps (NB) [KHZ00], and our Forest-ECOC (F-ECOC) with 3-embedded optimal trees. In the different variants of boosting we apply 50 iterations. We use Gentle Adaboost since it shown to outperform the other Adaboost variants in real applications [FHT00]. Finally, we apply FLDA as the base classifier for the Forest-ECOC.

Table A.3 shows the characteristics of the data used for the classification experiments, where #Training, #Test, #Features, and #Classes correspond to the number of training and test samples, number of features, and number of classes, respectively.

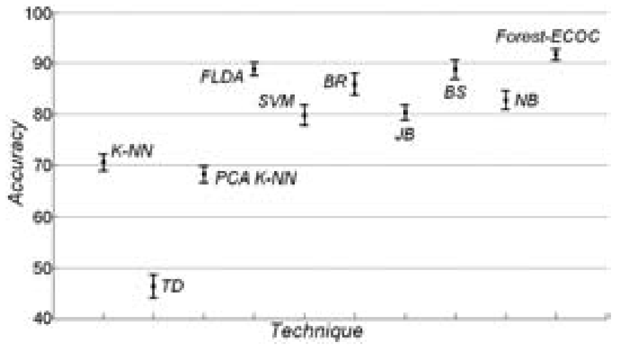
Table A.3
Characteristics of the databases used for classification. The training and test examples are equally distributed among the group classes.

Dataset	#Training examples	#Test examples	#Features	#Classes
Circular	750	200	1225	12
Speed	500	200	1681	7
Triangular	750	200	1716	12

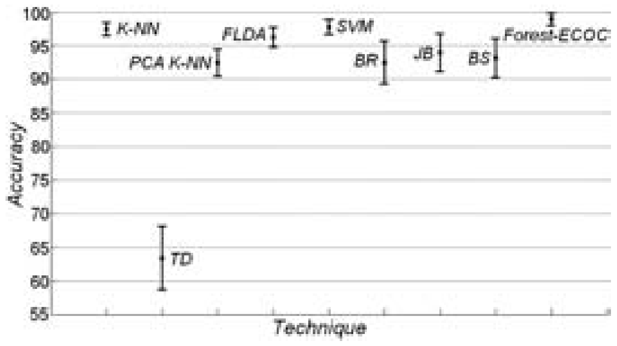
The classification results and confidence intervals are shown graphically in Fig. A.11 for the different groups. One can see that the Forest-ECOC using FLDA as a base classifier attains the highest accuracy in all cases. Nevertheless, for the circular and triangular signs the differences among classifiers are significatively different because of the high discriminability of these two groups. The speed group is a more difficult classification problem. In this case, the Forest-ECOC strategy obtains an accuracy upon 90%, outperforming the rest of classifiers.

Tree embedding analysis

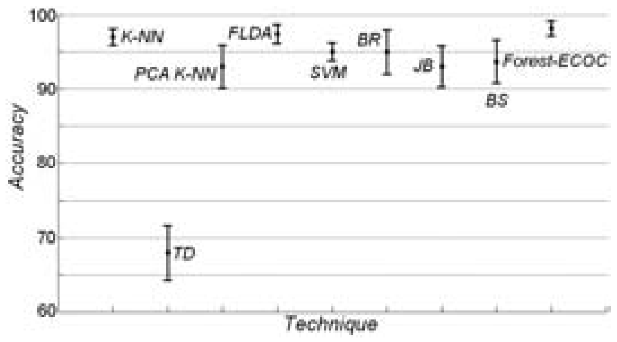
The training evolution of the Forest-ECOC at the previous experiment is shown in Fig. A.12 for the speed group. Each iteration of the figure shows the classification accuracy by embedding a new node (binary classifier) from each optimal tree in the Forest-ECOC matrix M. The three optimal trees are split by the dark vertical lines. The respective trees are shown in Fig. A.13. In the first generated tree of Fig. A.13, one can see that the most difficult partitions are reserved to the final classifiers of the tree. The next trees select the following best partitions of classifiers to avoid repeating classifiers. These classifiers learn sub-groups of classes from the same data, improving the classification results (Fig. A.12) by sharing their knowledge among classes.



(a)



(b)



(c)

Figure A.11: Classification results for the (a) Speed, (b) Circular, and (c) Triangular problems.

Model fitting classification

Finally, to test the performance of the classification step of the system, model fitting and Forest-ECOC classification are applied in a set of 200 regions of interests for each group. The regions of interest are obtained from the detection step. The results are

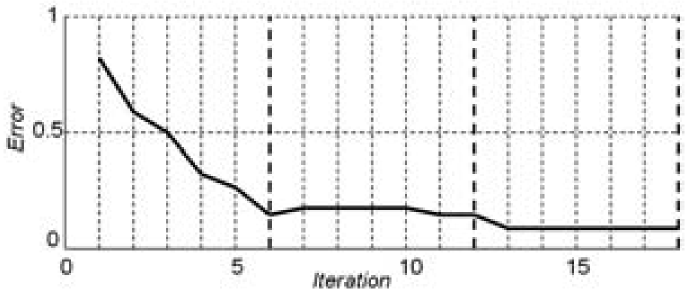


Figure A.12: Training process of Forest-ECOC embedding the first three optimal trees for the speed group.

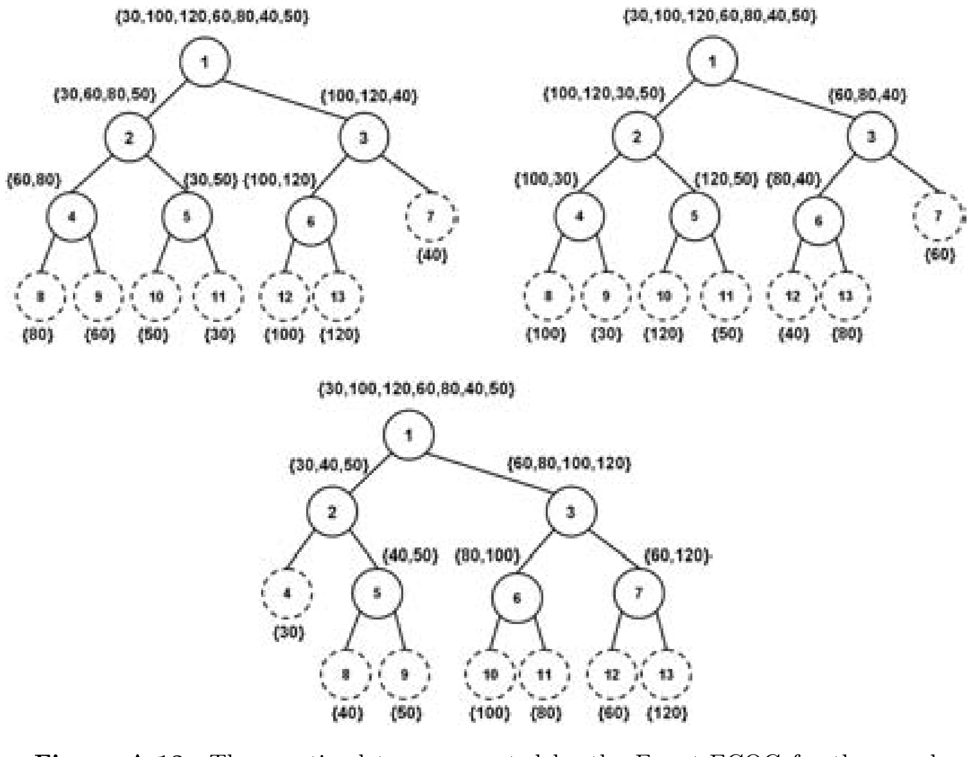


Figure A.13: Three optimal trees generated by the Forest-ECOC for the speed group.

shown in table A.4. One can see that for circular and speed signs the results are practically maintained from the previous experiments. For triangular signs, the accuracy is slightly decreased because of the effect of noise, variability of sign appearance, and resolution, that makes the Hough transform lose some sides of the triangular signs. Nevertheless, the final results are upon 90% in all cases.

Table A.4
Model fitting and classification results.

Recognition problem	Accuracy
Circular	98.00
Speed	91.50
Triangular	92.50

A.6 Conclusions

A complete system to deal with traffic sign recognition was developed (see Fig. A.14), and the performance of the whole system over a test set of 10.000 stereo-pairs of images, which correspond to $100Km$ of road, is calculated. The accuracy of the real traffic sign recognition system applying the detection and classification approaches jointly obtains a mean triangular sign reliability of $90.87 \pm 0.87\%$, and a circular sign reliability of $90.16 \pm 1.01\%$. In the detection stage, recognition fails are caused because of the background confusion (see Fig. A.9) and the high inter-class variability, whereas in the classification stage the errors are produced because of the poor resolution of the images.

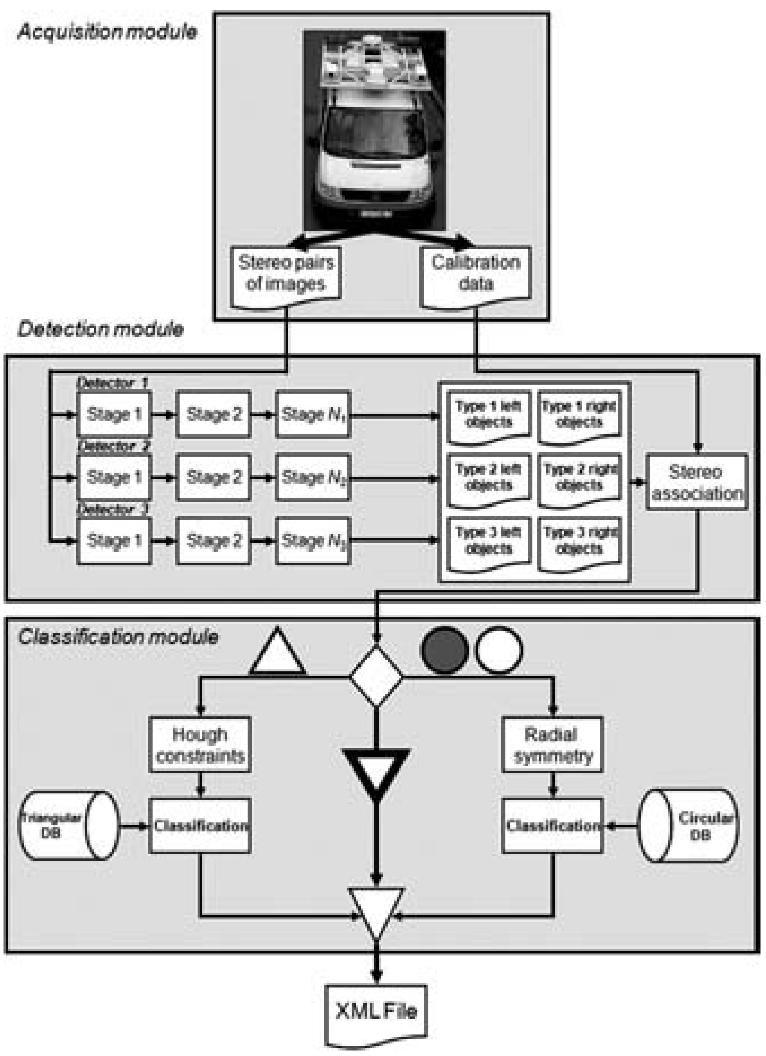


Figure A.14: The whole recognition system.

Appendix B

Feature Selection approach

One of the first approaches in order to face the learning time issue was based on feature selection, where the goal is to find a good subset of good features as a previous step of learning. This appendix describes a feature selection method based on the quadratic mutual information. This method allows to reuse part of the training time used in the first training process to speed up posterior training to update the detectors in front of samples changes.

B.1 Introduction

Feature selection methods are techniques to select a reduced subset of features from a normally very large set of features in order to solve a classification problem. This procedure can reduce not only the cost of classification by reducing the number of features, but in some cases it can also provide a better classification accuracy [JC82]. One of the most popular algorithms used on feature selection for classification is the Adaboost. As we introduce in Section 2.6.1, Boosting is a powerful learning concept that allows combining the performance of many simple classification functions to produce a strong classifier.

If we compare the number of available features with the number of features in the final detector, we can appreciate that just few of them are really used. The main problem of this method is the training time. With a training window size of 30×30 we can have more than 700.000 features, and training sets with thousands of images. In our tests applied to traffic signs (see Section A, a set of 1000 positive examples and a cascade goal false alarm ratio of 0.00001 spends a week to train. After the training stage, the cascade of 17 stages is formed by only 323 simple classifiers.

We think that this difference between the number of available and selected features is not a coincidence, and it means that most part of the features don't help on the classification problem. In this direction we bet for the mutual information between features and classes to select a priori a small set of features to solve the classification problem reducing the training time.

B.2 Mutual Information

We basically will follow the work of Torkkola in [TC00]. In this work he proposes a transformation instead of a features selection, but makes a very detailed study of all the problems and the methods to calculate the mutual information. We will assume that each feature X is an univariate random variable and C a discrete-valued random variable representing the class labels. In following equations, uppercase P will denote a probability and lowercase p a probability density. Given a sample, the entropy or uncertainty of the class label, making use of Shannon's definition, can be expressed in terms of class prior probabilities.

$$H(C) = -\sum_c P(c) log(P(c)) \tag{B.1}$$

Once we have observed a feature value x, the uncertainty of the class label is expressed as:

$$H(C|X) = -\int_x p(x) \left(\sum_c p(c|x) log(p(c|x)) \right) dx \tag{B.2}$$

The amount by which the class uncertainty is reduced, after having observed the feature vector x, is the mutual information, which can be written as:

$$I(C, X) = \sum_c \int_x p(c, x) log \frac{p(c, x)}{P(c)p(x)} dx \tag{B.3}$$

The practical estimation of the mutual information from data based on expression (B.3) is difficult because the good estimation of the probability density function of a continuous variable is not easy. To solve this problem, in the following sections we describe a method to calculate a mutual information measure based on a reformulation of the entropy concept and a density estimator method. The result is a formulation of mutual information in terms of discrete sums.

B.3 Parzen density estimator

The Parzen window method [Par62] is a non-parametric method to estimate the probability density function. This method involves placing a kernel function on top of each sample and evaluate the density as a sum of kernels. The Gaussian kernel is defined as:

$$G(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-\mu^2}{2\sigma^2}} \tag{B.4}$$

Now, for two kernels, we can write:

$$\int_x G(x - \mu_1, \sigma_1^2) G(x - \mu_2, \sigma_2^2) = G(\mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2) \tag{B.5}$$

Thus, the convolution of two Gaussians centered at μ_1 and μ_2 is a Gaussian centered at $\mu_1 - \mu_2$, with a variance equal to the sum of variances. Assume now that the density of X is estimated as a sum of Gaussians centered at a sample x_i. This is the Parzen density estimation:

$$p(x) = \frac{1}{N} \sum_{i=1}^{N} G(x - x_i, \sigma^2) \tag{B.6}$$

where N is the number of samples.

B.4 Renyi's Entropy

Renyi's entropy is a more general formulation than Shannon entropy. In the general theory of means [Ren61], the mean of the real numbers $x_1, ..., x_N$ with positive weighting (not necessarily probabilities) $p_1, ..., p_N$ has the form:

$$\bar{x} = \varphi^{-1}\left(\sum_{k=1}^{N} p_k \varphi(x_k)\right) \tag{B.7}$$

where $\varphi(x)$ is a Kolmogorov-Nagumo function, which is an arbitrary continuous and strictly monotonic function defined on the real numbers. In general, an entropy measure H obeys the relation:

$$H = \varphi^{-1}\left(\sum_{k=1}^{N} p_k \varphi(I(p_k))\right) \tag{B.8}$$

where $I(p_k) = -log(p_k)$ is Hartley's information measure [Har28]. In order to be an information measure, $\varphi(.)$ can not be arbitrary since information is "additive". To meet additivity condition, $\varphi(.)$ can be either $\varphi(x) = x$ or $\varphi(x) = 2^{(1-\alpha)x}$. If $\varphi(x) = x$ is selected, (B.8) will become Shannon's entropy. For $\varphi(x) = 2^{(1-\alpha)x}$ Renyi's entropy of order α is obtained [Ren76], which we will denote by H_{R_α}

$$H_{R_\alpha} = \frac{1}{1-\alpha} log \left(\sum_{k=1}^{N} p_k^\alpha\right) \qquad \alpha > 0, \alpha \neq 1 \tag{B.9}$$

In fact, Renyi's entropy of order α will compute interactions among α-tuples of samples, providing even more information about the complex structure of the data set[PFX00]. When $\alpha = 2$, (B.9) is called quadratic entropy due to the quadratic form on the probability. For a discrete variable C and a continuous variable X, the quadratic Renyi entropy H_{R_2} is defined as [Ren61]:

$$H_{R_2}(C) = -log \sum_{c} p(c)^2 \qquad H_{R_2}(X) = -log \int_{x} p(x)^2 dx \tag{B.10}$$

Note that Renyi's quadratic entropy involves the use of the square of the PDF. An important observation is that this alternate definition of entropy is equivalent to Shannon's entropy for the goal of entropy maximization [Kap94]. Then, it follows that the quadratic Renyi's entropy in (B.10) equals [TC00]

$$\begin{aligned} H_{R_2}(X) &= -log \int_x p(x)^2 dx \\ &= -log\tfrac{1}{N^2} \int_x \left(\sum_{k=1}^{N} \sum_{j=1}^{N} G(x - x_k, \sigma^2) G(x - x_j, \sigma^2) \right) dx \\ &= -log\tfrac{1}{N^2} \sum_{k=1}^{N} \sum_{j=1}^{N} G(x_k - x_j, 2\sigma^2) \end{aligned} \tag{B.11}$$

Thus, Renyi quadratic entropy can be estimated as a sum of local interactions, as defined by the kernel, over all pairs of samples.

B.5 Information potentials

Assume that we have J_p samples for each class c_p. Then, the class prior probabilities are $P(c_p) = J_p/N$, with $\sum_{p=1}^{N_c} Jp = N$. Now we will use different notations for the samples of data X. A sample is written with a single subscript x_i when its class is irrelevant. If the class is relevant, we will write x_{pj}, where p is the class index and j the within-class index. N_c is the number of classes.

The density of each class c_p, as a Parzen estimate using the Gaussian kernel of width σ, is written as:

$$p(x|c_p) = \frac{1}{J_p} \sum_{j=1}^{J_p} G(x - x_{pj}, \sigma^2) \tag{B.12}$$

Using the definition of joint density $p(c, x) = p(x|c)P(c)$, we have

$$p(c_p, x) = \frac{1}{N} \sum_{j=1}^{J_p} G(x - x_{pj}, \sigma^2), p = 1, ..., N_c \tag{B.13}$$

Finally, using that the density of all data is $p(x) = \sum_c p(c, x)$, we can write

$$p(x) = \frac{1}{N} \sum_{p=1}^{N_c} \sum_{j=1}^{J_p} G(x - x_{pj}, \sigma^2) = \frac{1}{N} \sum_{i=1}^{N} G(x - x_i, \sigma^2) \tag{B.14}$$

Using the quadratic entropy in the calculus of mutual information, we can speak of quadratic mutual information, denoted by I_T. With continuous-valued X and discrete C, the definition of the quadratic mutual information can be written as [PFX00]:

$$\begin{aligned} & I_T(C, X) = V_{IN} + V_{ALL} - 2V_{BTW} \\ & where \quad \begin{cases} V_{IN} \equiv \sum_c \int_x p(c, x)^2 dx \\ V_{ALL} \equiv \sum_c \int_x P(c)^2 p(x)^2 dx \\ V_{BTW} \equiv \sum_c \int_x p(c, x) P(c) p(x) dx \end{cases} \end{aligned} \tag{B.15}$$

Using a set of samples $\{x_i\}$, combining the equations (B.13),(B.14) and (B.15), and making use of (B.5) and (B.11), we get:

$$V_{IN}(\{c_i, x_i\}) = \sum_c \int_x p(c, x)^2 dx = \frac{1}{N^2} \sum_{p=1}^{N_c} \sum_{k=1}^{J_p} \sum_{l=1}^{J_p} G(x_{pk} - x_{pl}, 2\sigma^2) \tag{B.16}$$

$$V_{ALL}(\{c_i, x_i\}) = \sum_c \int_x P(c)^2 p(x)^2 dx = \frac{1}{N^2}\left(\sum_{p=1}^{N_c}\left(\frac{J_p}{N}\right)^2\right)\sum_{k=1}^{N}\sum_{l=1}^{N} G(x_k - x_l, 2\sigma^2) \tag{B.17}$$

$$V_{BTW}(\{c_i, x_i\}) = \sum_c \int_x p(c,x)P(c)p(x)dx = \frac{1}{N^2}\sum_{p=1}^{N_c}\frac{J_p}{N}\sum_{j=1}^{J_p}\sum_{k=1}^{N} G(x_{pj} - x_k, 2\sigma^2) \tag{B.18}$$

These kinds of quantities can be called "information potentials" in analogy to physical particles [PFX00]. In the next section we present a method to calculate the value of the σ used in these equations.

B.6 Sigma estimation

The correct selection of the sigma value has a capital importance for the correctness of the final mutual information values. This parameter depends on the data, and can be viewed as the window width in the Parzen method. In Fig. B.1 is showed the effect of the variation of this value in the estimated probability density function. In

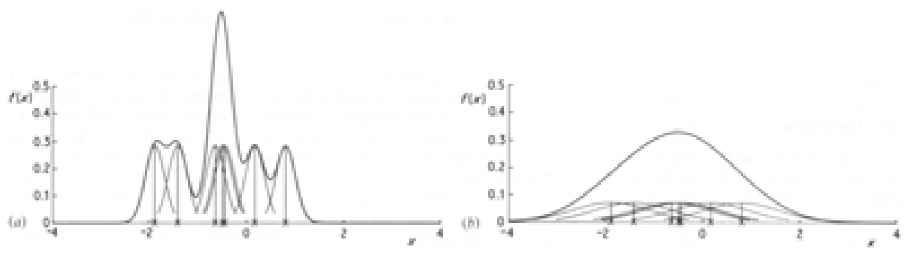


Figure B.1: From [Sil86], Kernel estimation showing individual kernels. Windows widths: (a) 0.2;(b)0.8.

[Sil86], Silveman develops a set of equations in order to select a correct value in the case of Gaussian kernel, minimizing the mean integrated square error. The resultant equation is:

$$\sigma = 0.9An^{-\frac{1}{5}} \tag{B.19}$$

where $A = min(standard\ deviation, interquartile\ range/1.34)$ and n is the number of samples used for the estimation.

B.7 Results

First we briefly presents some statistics over the detection step to justify the necessity to use some methods to reuse the time spent training the system. The data and underlying structure is taken from Section A, were traffic signs are divided in five classes (yield, danger, prohibition, command and kilometric points). For each class we trained a different detector, which is a cascade of detectors trained using Adaboost.

In table B.1 we show the mean number of features per stage, and one can see that is a number really smaller than the original 700.000 features of the features set. The results obtained when we apply our detectors to the test are quite low due to the difference in the orientation, illumination and/or kind of the signs. We want to add the failed signs to the train set to improve the detectors. Using Viola & Jones approach, it means to spend a week for each detector, and here is where we want to introduce the feature selection methods, that can reduce drastically this time.

Table B.1

Training set size and performance of each detector after analyzing 9510 frames of 1020×1024. All the detectors are trained on a size of 30×30 except the kilometric points trained at 24×24. HR is the obtained hit ratio and FA the false alarm ratio.

Sign type	#Training Samples	Mean features per stage	#Signs	HR	FA
Yield	425	9.176	179	93.08%	1060/179=5.92
Danger	545	12.125	385	89.59%	854/385=2.21
Prohibition	993	19	481	83.36%	371/481=0.77
Command	356	11.667	115	70.88%	1382/115=12.01
Km points	218	8	148	76.99%	2928/148=19.78

To measure the influence of the mutual information to the boosting process, we compare the convergence speed of the AdaBoost selecting small sets of features. The following tests are programmed in Matlab, using a sampled features set of 8000 rectangular features and a set of 400 samples (50% positive and 50% negative). We use the Discrete Adaboost algorithm, fixing the number of iterations at 100.

Using all the features the training process spend more than four hours and half, and it converge at iteration 13. If we select only the 100 features with the higher mutual information, the training time is reduced to only five minutes, and it converges at iteration 39. Using only the 100 worst features, and maintaining the training time, the convergence of the Adaboost is delayed to the iteration 83 (see Fig. B.2). It is important to emphasis that at the end of the training process, in all cases the detector obtain the same detection rates.

Finally, we compare the effect of the number of features over the convergence speed. Using the same training set as before, we train a classifier using the N features with better mutual information. The results are in Fig. B.3. We can see that from a certain number of features, to add more features has a moderate effect over the convergence speed. Our interpretation is that from a certain number of features, the new features do not apport important information for the classification process.

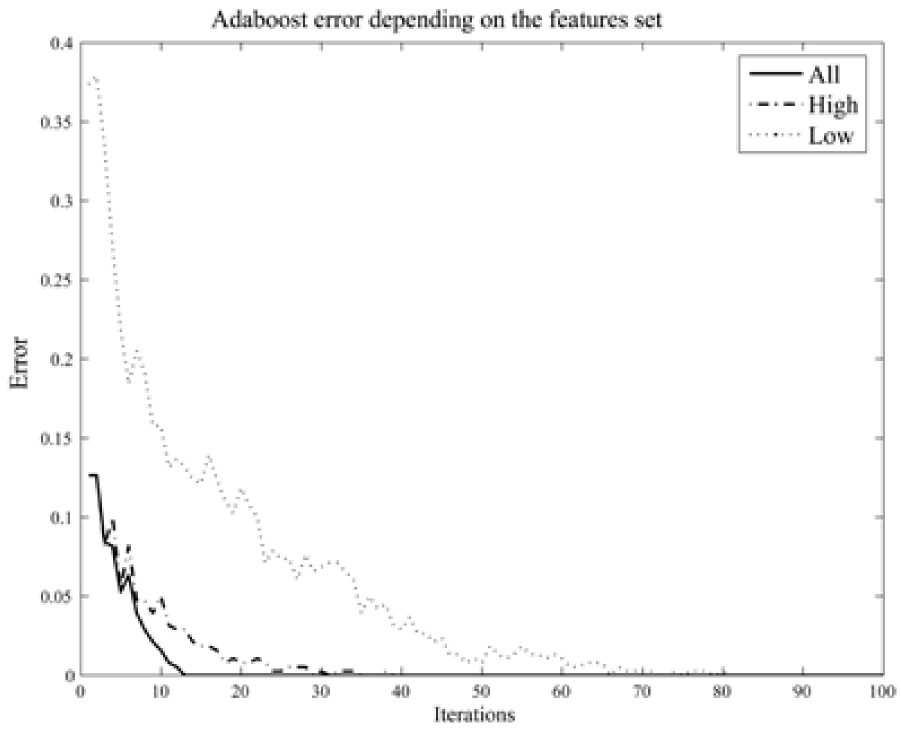


Figure B.2: Convergence of the Adaboost depending on the features set. First using all features, then only the 100 with the higher MI and finally the 100 with lower MI.

B.8 Conclusions and future work

The convergence speed shows that the mutual information between features and class labels have a direct relationship with the convergence speed of the AdaBoost algorithm.

To calculate the mutual information between each feature and the class labels is too expensive in time to be calculated each time. The main idea is use the first samples set to calculate the mutual information and select a small subset of features. Then each time that we add new samples to our training set, we will repeat the training process only with this selected features, and it will reduce drastically the time used to maintain all detectors up to date.

In this first approach, we only select the features with higher mutual information, but is logical to use the mutual information to select also the features with the minimal mutual information with the other features, to eliminate redundant features.

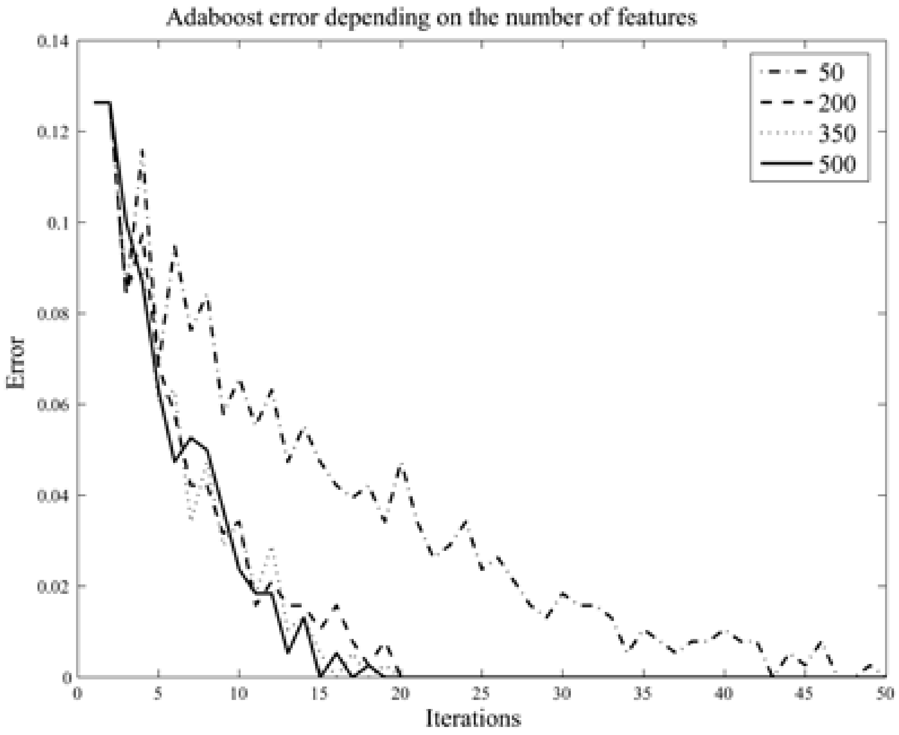


Figure B.3: Convergence of the Adaboost depending on the number of features. Features are sorted by their mutual information value and the features set are the N features with the higher MI.

Appendix C

Problems and Image Databases

In order to verify the usefulness of the presented methodology we have selected different real world object detection problems, using for each one a public database to facilitate the reproduction and comparison of the results. In some cases, the databases contain both, the object image and background images to use as negative samples, and when the background images are not given, we create the negative samples from the *Corel Photo Libraries* [Cor96](see Section C.6). In the following, the databases are described in order to give a general vision of each one of the problems.

C.1 Face detection

Detecting human faces in images and video sequences is an important task for many applications. Enhanced video surveillance and security related applications, in particular, are closely related with human face identification where the task of face detection becomes a major requisite to deliver efficient and robust performance under challenging conditions. The wide applicability of face detection made of this problem one of the most studied in the computer vision field, and it is reflected in the amount of published works for this topic.

In order to compare the reliability of our methods in this topic, we use the MIT-CBCL face database [CBC]. This database contains 2.429 faces and 4.548 non-faces at a low resolution of 19×19 pixels. The images correspond to the inner part of frontal faces with several illumination changes. In Figure C.1 the mean image from the positive examples is shown. Notice that in the mean image we are able to identify a face, therefore, the images in general are aligned and they share a structure.

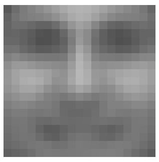


Figure C.1: Mean image of positive images in the face detection problem.

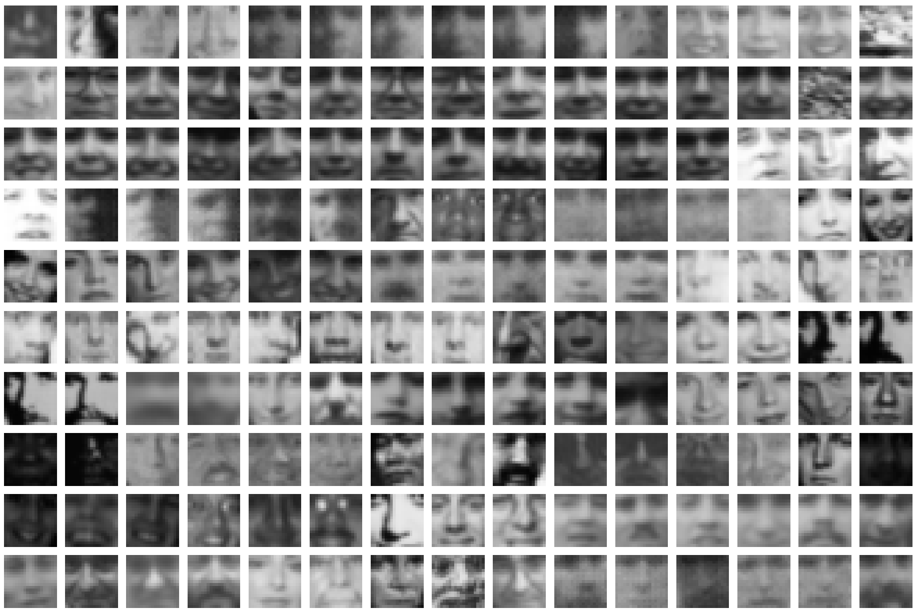


Figure C.2: Positive samples for the face detection problem.

Figure C.3: Negative samples for the face detection problem.

To analyze the variability of images in the database, positive images are divided in clusters using a K-mean approach, and not empty clusters are shown in Fig. C.4.

Notice that only two clusters appears, with the lighter and darker regions in an

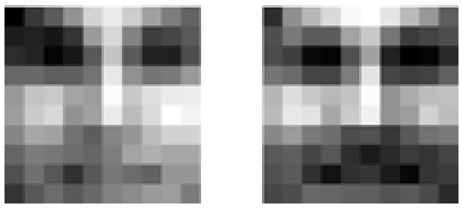


Figure C.4: Mean image for each cluster of the face detect database.

opposite order.

C.2 Pedestrian detection

The ability to reliably detect pedestrians in real-world images is interesting for a variety of applications, such as video surveillance or automatic driver-assistance systems in vehicles, becoming an essential and significant task in any intelligent video surveillance system, as it provides the fundamental information for semantic understanding of the video sequences. From a complexity point of view, pedestrians are one of the most challenging categories for object detection. A large variability in their local and global appearance is caused by various types and styles of clothing, so that only few local regions are really characteristic for the entire category. Moreover, the global shape undergoes a large range of transformations due to the variety of possible articulations and a multitude of occluding accessories.

We use the INRIA Person Dataset[1], with 2.924 images divided into 924 pedestrian instances and 2.000 background images.

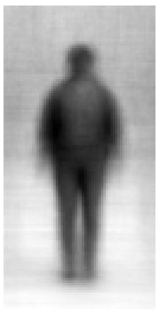


Figure C.5: Mean image of positive images in the pedestrian detection problem.

The variability of images in the database it analyze using the clusters obtained with the K-Mean algorithm. Not empty clusters are shown in Fig. C.7. Notice that main differences consists on the contrast between the pedestrian and the background, and are contained in some clusters.

[1]`pascal.inrialpes.fr/data/human/`

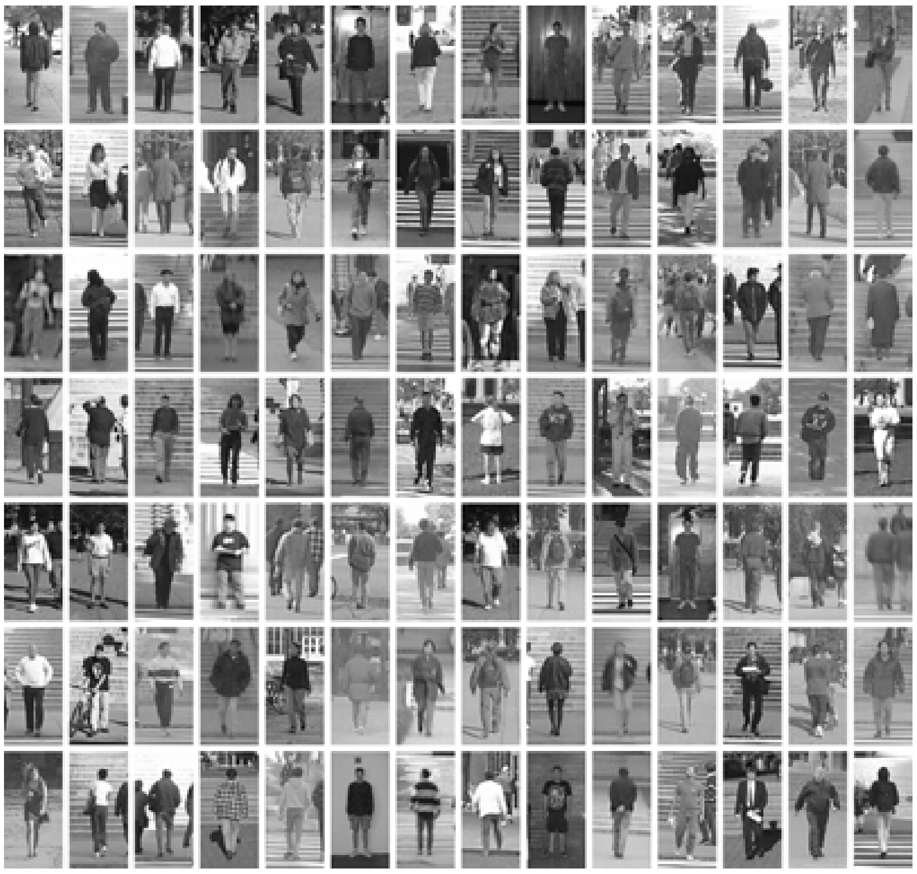


Figure C.6: Positive samples for the pedestrians detection problem.

C.3 Car detection

Cars detection is a considerably more difficult problem than detecting faces or pedestrians. In human faces, semi-rigid structure simplify the problem, where the localization of face components does not vary much between samples. In the case of cars, although they have a semi-rigid structure, that structure will vary more between samples. Shapes, colors and structure of cars have been designed in order to create different products with an added value to the customer. Moreover, cars suffer from a large variability due to the point of view, while faces has far fewer degrees of freedom, because only frontal views, side profiles, and any pose in between are of general interest. This restriction reduces the intra-instance variability due to viewing conditions.

We use the UIUC cars database [AAR], with a total of 1.050 images containing 550 instances of lateral views of different cars in urban scenes and 500 images of background.

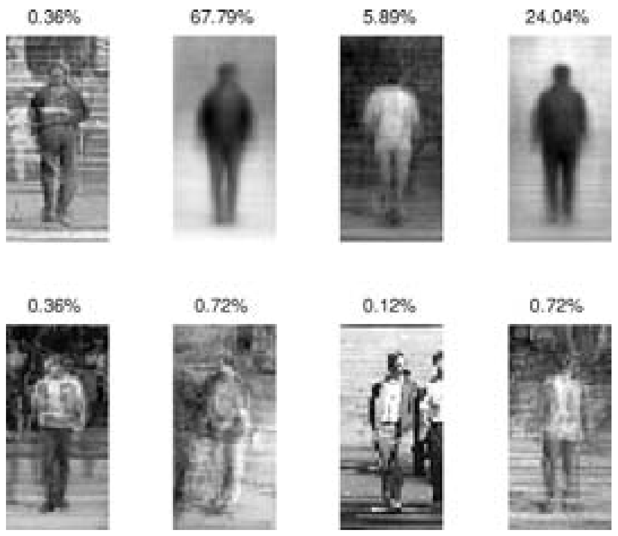


Figure C.7: Mean image for each cluster of the INRIA Person dataset. Numbers over clusters are the percentage of images summarized on each cluster.

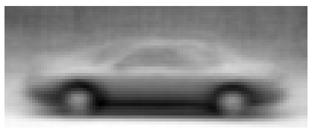


Figure C.8: Mean image of positive images in the car detection problem.

The variability of images in the database it analyze using the clusters obtained with the K-Mean algorithm. Not empty clusters are shown in Fig. C.11. Notice that we find more variability on car images than in the case of faces, and a large number of clusters in necessary to represent all the data. Analyzing these clusters, we detect cars with lighter colors than others and in opposite directions.

C.4 Text detection

Text detection or location is a problem widely studied in the field of document analysis. The majority of works that face this problem performs this task over a document, in which case they usually can use the structure of the document in order to detect the regions with text. Recently, a growing interest on text detection on video data motivates new methodologies to solve this problem. However, the detection of text regions in uncontrolled environments has not yet widely studied, and is considered a hard task due to the huge variance of text regions.

In order to evaluate the performance of our methods in this problem, we use the text location dataset from the *7th International Conference on Document Analysis*

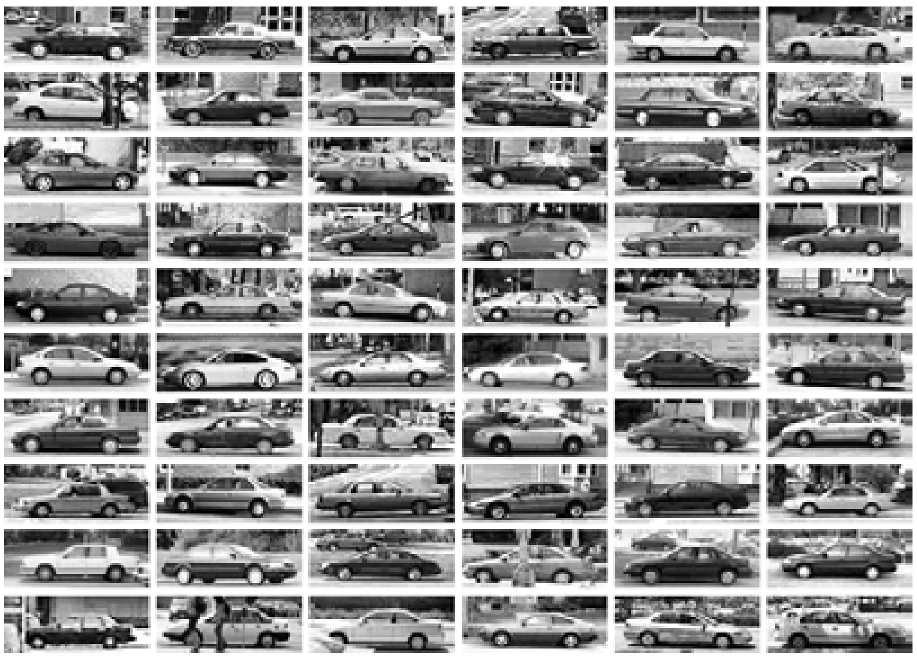


Figure C.9: Positive samples for the car detection problem.

and Recognition (ICDAR03)[2]. The organizers of the ICDAR03 proposed a challenge on this topic, providing a large amount of text regions in a wide variety of surfaces, illumination conditions, and type fonts (see Figure C.12).

Notice that in contrast with the previous problems where the objects can be normalized to have the same size, in this case we can only normalize the height, because the width depends on the length of the text. Since in our work we assume a learning window where the features are located, we split the text images into overlapped subregions with the same size. An example for this process is schematized on Figure C.13, and the resulting regions are shown in Figure C.14.

Following the same process as in the previous problems, we create the mean image from the generated images. Notice in Figure C.15, that although the images have an structure, we cannot recognize a text in the mean image.

Since no negative examples are provided, negative samples are generated as explained in section C.6.

The intra-class variability is analyzed using a clustering process over the positive samples. The mean image for each cluster is shown in Fig. C.16.

[2]`algoval.essex.ac.uk/icdar/TextLocating.html`

Figure C.10: Negative samples for the car detection problem.

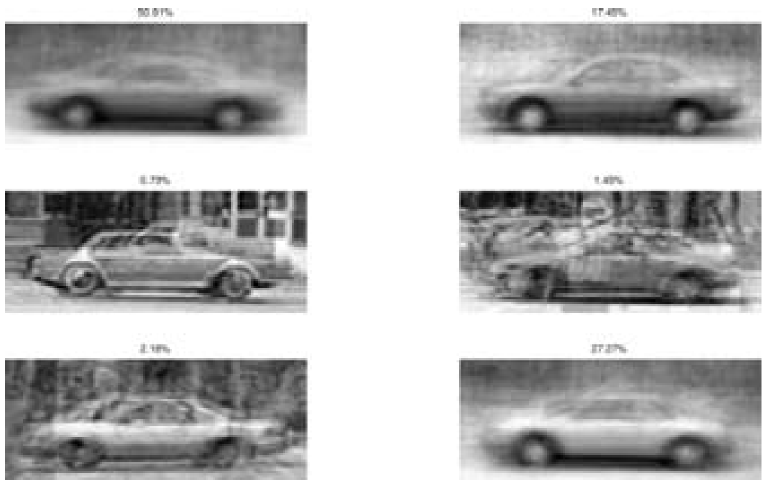


Figure C.11: Mean image for each cluster of the UIUC Cars database.

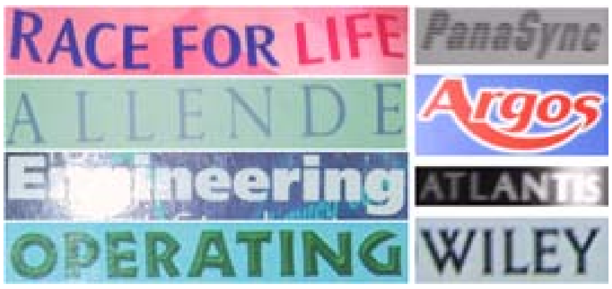


Figure C.12: Examples of the text regions in the ICDAR03 text location challenge.

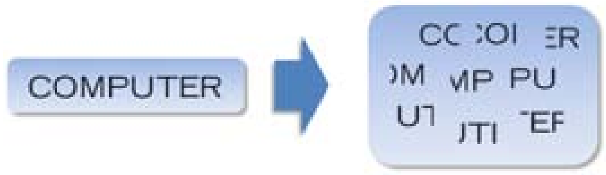


Figure C.13: Example of the text splitting process.

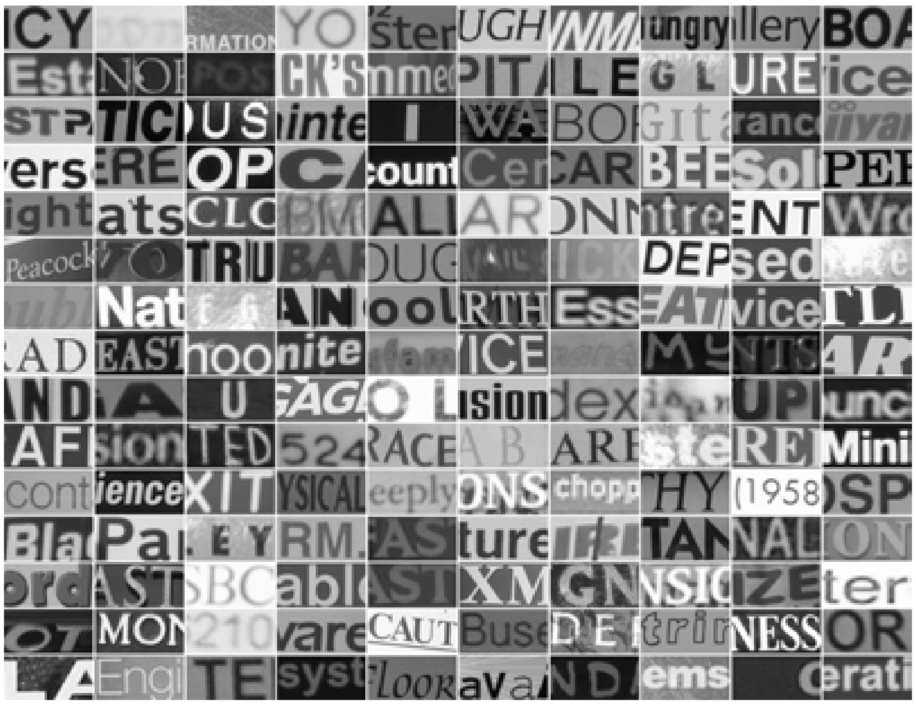


Figure C.14: Positive samples for the text detection problem.

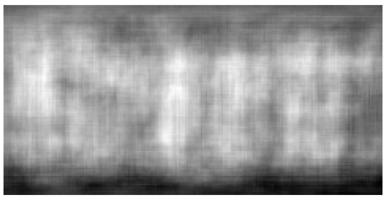


Figure C.15: Mean image of positive images in the text detection problem.

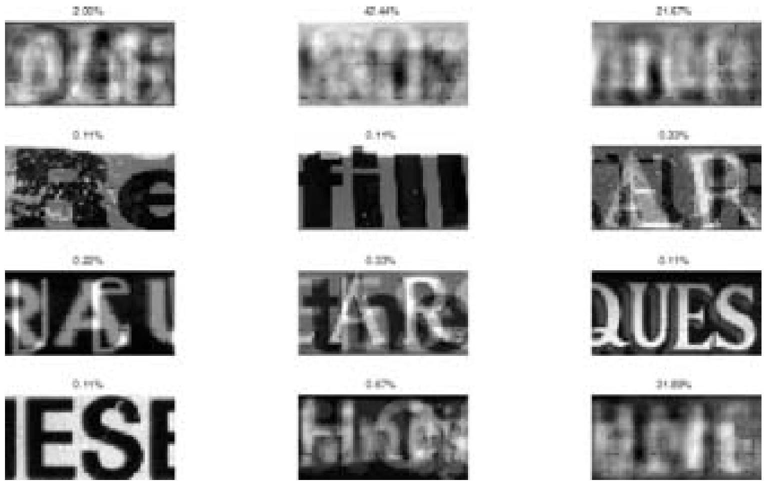


Figure C.16: Mean image for each cluster of the ICDAR'03 Text database.

C.5 Traffic Sign detection

In this case we use real images acquired in the context of a mobile mapping project provided by the ICC[3]. The database consists on 1.000 images containing a traffic sign and 3.000 background images.

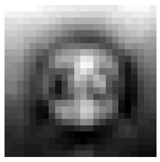


Figure C.17: Mean image of positive images in the traffic sign detection problem.

The intra-class variability is analyzed using a clustering process over the positive samples. The mean image for each cluster is shown in Fig. C.19. Notice that since most part of traffic signs have a white center, this is the predominant color in most clusters. Moreover, different types are contained in different clusters.

[3]Institut Cartogràfic de Catalunya. `www.icc.es`

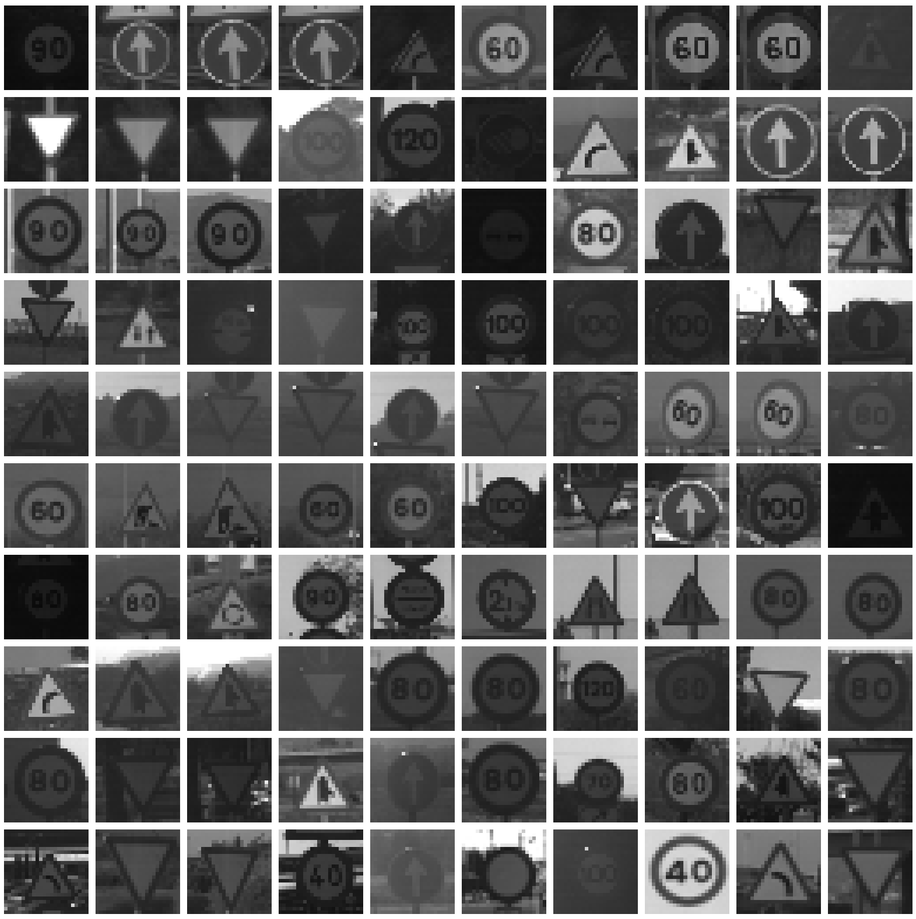


Figure C.18: Positive samples for the traffic sign detection problem.

C.6 Negative samples generation

Negative samples are generated from the *Corel Photo Libraries* [Cor96]. This is a library of 68.040 photo images from various categories. As can be seen in Fig. C.20 there are images from people, landscapes, known tourist locations, plants, etc....

In order to create the negative samples from the Corel Photo Libraries with a large diversity, some random processes are performed. First, an image is selected randomly from the whole dataset. Once we have an image, we select a random region at a random scale. This region is cut and resized to the target size. This process is repeated for each negative sample.

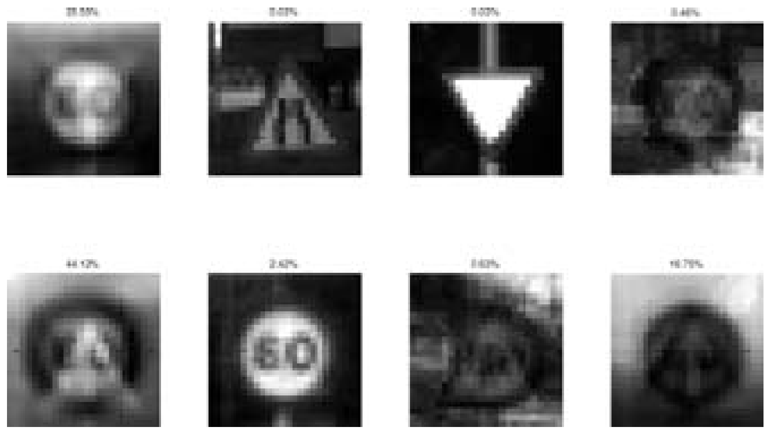


Figure C.19: Mean image for each cluster of the Traffic Sign database.

Figure C.20: Samples from the Corel Photo Libraries.

Appendix D

Statistical analysis of results

This appendix describes the methodology used in order to compare the results obtained in the different experiments of the thesis. In [Dem06], Demšar performs a study of the validation schemes used in the works published in the International Conferences on Machine Learning between 1999 and 2003, pointing up the main validation errors and wrong assumptions. As a result of this study, Demšar concludes that there is no established procedure for comparing classifiers over multiple data sets. Various researchers adopt different statistical and common-sense techniques to decide whether the differences between the algorithms are real or random. Finally, the author describes a methodology to compare a set of methods over different data sets. In order to compare our results in a coherent manner, the analysis of the experimental results is analyzed using the methodology proposed by Demšar. This methodology is summarized in this appendix, given the main formulation and a step-by-step methodology to extract the conclusions.

The goal is to compare a set of k learning algorithms (or configurations for a certain algorithm) over N data sets. Let c_i^j be the performance score of the $j-$th algorithm on the $i-$th data set. Our goal is to decide whether, based on the values c_i^j, the algorithms are statistically significantly different. In addition, when we have more than two algorithms, we are also interested on which of the algorithms are the particular algorithms that differ in performance. In the case of multiple repetitions of the experiments, we can only take into account the variance values σ_i^j if all the observations are independent. In general, the standard validation methodologies do not accomplish this assumption, that is, most of the observations are shared between the different repetitions of the experiment (i.e in a $K-$fold cross validation, at least $K-2$ groups of samples are shared between two consecutive learning cycles). Since in our experiments never can ensure independence between observations, in following the variances are deprecated, and only the mean value is considered to perform the statistical analysis.

When we are working on significance testing, the first step is to determine whereas the null hypothesis, which states that all the algorithms are equivalent, can be rejected. In this context, there are two kinds of errors that can appear during the statistical analysis:

Type I: A true null hypothesis is incorrectly rejected. The probability of a *Type I* error is commonly designated by α and is called the *Type I error rate.*

Type II: A false null hypothesis is failed to be rejected. The probability of a *Type II* error is commonly designated by β and is called the *Type II error rate.* A Type II error is only an error in the sense that an opportunity to reject the null hypothesis correctly was lost. It is not an error in the sense that an incorrect conclusion was drawn since no conclusion is drawn when the null hypothesis is not rejected.

The issue of multiple hypothesis testing is a well-known statistical problem. The usual goal is to control the family-wise error, the probability of making at least one *Type I* error in any of the comparisons.

Let r_i^j be the rank of the $j-$th of k algorithms on the $i-$th of N data sets. The Friedman test compares the average ranks of algorithms, $R_j = \frac{1}{N}\sum_i r_i^j$. Under the null-hypothesis, all the algorithms are equivalent, and so their average ranks R_j are equal, the Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{D.1}$$

is distributed according to χ_F^2 with $k-1$ degrees of freedom when N and k are big enough (i.e $N > 10$ and $k > 5$). For a small number of algorithms and data sets, exact critical values have been computed [Zar98, She00].

In [ID80], Iman and Davenport showed that Friedman's χ_F^2 is undesirably conservative and derived a better statistic

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2} \tag{D.2}$$

which is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom. The null-hypothesis will be rejected only if F_F is smaller than the critical value of the F-distribution for a given confidence value α. The critical values can be found in any statistics book, and a representation is shown in Table D.1 and Table D.2.

At this point, if the null-hypothesis is rejected, we can proceed with a post-hoc test, otherwise, the methods are not statistically different, therefore, or they have equal performance or we need more data sets in order to reject the null-hypothesis. There are two possible scenario: The Nemenyi test [Nem63] is used when all classifiers are compared to each other. The performance of two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference

$$CD = q_\alpha\sqrt{\frac{k(k+1)}{6N}} \tag{D.3}$$

where critical values q_α are based on the Studentized range statistic divided by $\sqrt{2}$ (see Table D.3).

Table D.1
F Distribution critical values for $\alpha = 0.1$

$v_1 \backslash v_2$	**1**	**2**	**3**	**4**	**5**	**7**	**10**	**15**	**20**
1	161.45	199.50	215.71	224.58	230.16	236.77	241.88	245.95	248.01
2	18.513	19.000	19.164	19.247	19.296	19.353	19.396	19.429	19.446
3	10.128	9.5522	9.2766	9.1172	9.0135	8.8867	8.7855	8.7028	8.6602
4	7.7086	6.9443	6.5915	6.3882	6.2560	6.0942	5.9644	5.8579	5.8026
5	6.6078	5.7862	5.4095	5.1922	5.0504	4.8759	4.7351	4.6187	4.5582
7	5.5914	4.7375	4.3469	4.1202	3.9715	3.7871	3.6366	3.5108	3.4445
10	4.9645	4.1028	3.7082	3.4780	3.3259	3.1354	2.9782	2.8450	2.7741
15	4.5431	3.6823	3.2874	3.0556	2.9013	2.7066	2.5437	2.4035	2.3275
20	4.3512	3.4928	3.0983	2.8660	2.7109	2.5140	2.3479	2.2032	2.1241
30	4.1709	3.3159	2.9223	2.6896	2.5336	2.3343	2.1646	2.0149	1.9317

Table D.2
F Distribution critical values for $\alpha = 0.05$

$v_1 \backslash v_2$	**1**	**2**	**3**	**4**	**5**	**7**	**10**	**15**	**20**
1	39.864	49.500	53.593	55.833	57.240	58.906	60.195	61.220	61.740
2	8.5264	8.9999	9.1618	9.2434	9.2926	9.3491	9.3915	9.4248	9.4413
3	5.5384	5.4624	5.3907	5.3426	5.3092	5.2661	5.2304	5.2003	5.1845
4	4.5448	4.3245	4.1909	4.1073	4.0505	3.9790	3.9198	3.8704	3.8443
5	4.0605	3.7798	3.6194	3.5202	3.4530	3.3679	3.2974	3.2379	3.2067
7	3.5895	3.2575	3.0740	2.9605	2.8833	2.7850	2.7025	2.6322	2.5947
10	3.2850	2.9244	2.7277	2.6054	2.5216	2.4139	2.3226	2.2434	2.2007
15	3.0731	2.6951	2.4898	2.3615	2.2729	2.1582	2.0593	1.9722	1.9243
20	2.9746	2.5893	2.3801	2.2490	2.1582	2.0397	1.9368	1.8450	1.7939
30	2.8808	2.4887	2.2761	2.1423	2.0493	1.9269	1.8195	1.7222	1.6674

Table D.3
Critical values for the two-tailed Nemenyi test

# methods	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$q_{0.05}$	1.960	2.343	2.569	2.728	2.850	2.949	3.031	3.102	3.164
$q_{0.10}$	1.645	2.052	2.291	2.459	2.589	2.693	2.780	2.855	2.920

In those cases when one of the methods is used as control method, comparing the performance of the rest against this one, we can instead of the Nemenyi test use one of the general procedures for controlling the family-wise error in multiple hypothesis testing, such as the Bonferroni correction or similar procedures. Although these methods are generally conservative and can have little power, they are in this specific case more powerful than the Nemenyi test, since the latter adjusts the critical value for making $\frac{k(k-1)}{2}$ comparisons while when comparing with a control we only make $k-1$ comparisons. The test statistics for comparing the $i-$th and $j-$th classifier using these methods is

$$z = \frac{(Ri - Rj)}{\sqrt{\frac{k(k+1)}{6N}}} \tag{D.4}$$

The z value is used to find the corresponding probability from the table of normal distribution, which is then compared with an appropriate α. The tests differ in the way they adjust the value of α to compensate for multiple comparisons.

The Bonferroni-Dunn test [Dun61] controls the family-wise error rate by dividing α by the number of performed comparisons $k-1$. The alternative way to compute the same test is to calculate the CD using Equation D.3, but now using the critical values for $\frac{\alpha}{(k-1)}$ (see Table D.4). The comparison between the tables for Nemenyi's and Bonferroni-Dunn's test shows that the power of the post-hoc test is much greater when all classifiers are compared only to a control classifier and not between themselves. We thus should not make pairwise comparisons when we in fact only test whether a newly proposed method is better than the existing ones.

Table D.4
Critical values for the two-tailed Bonferroni-Dunn test. The number of methods include the control method

# methods	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$q_{0.05}$	1.960	2.241	2.394	2.498	2.576	2.638	2.690	2.724	2.773
$q_{0.10}$	1.645	1.960	2.128	2.241	2.326	2.394	2.450	2.498	2.539

Bibliography

[AAR] S. Agarwal, A. Awan, and D. Roth. UIUC cars database.

[ABB$^+$04] R. Alamús, A. Baron, E. Bosch, J. Casacuberta, J. Miranda, M. Pla, S. Sànchez, A. Serra, and J. Talaya. On the accuray and performance of the geomobil system. In *"International Society for Photogrammetry and Remote Sensing (ISPRS '04)"*, "Istambul, Turkey", "July" 2004.

[AI87] H. Akatsuka and S. Imai. Road signposts recognition system. In *SAE vehicle highway infrastructure: safety compatibility*, pages 189–196, 1987.

[AMP87] Y.S. Abu-Mostafa and D. Pslatis. Optical neural computing. In A. Prieditis and S. Russel, editors, *Scientific American*, volume 256, pages 66–73, 1987.

[ASS02] E. Allwein, R. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2002.

[Bak85] J.E. Baker. Adaptive selection methods for genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 101–111, Mahwah, NJ, USA, 1985. Lawrence Erlbaum Associates, Inc.

[Bar54] N.A. Barricelli. Esempi numerici di processi di evoluzione. *Methodos*, pages 45–68, 1954.

[Bau00] A. Baumberg. Reliable feature matching across widely separated views. *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, 1:774–781, 2000.

[BC95] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. In A. Prieditis and S. Russel, editors, *The Int. Conf. on Machine Learning 1995*, pages 38–46, San Mateo, CA, 1995. Morgan Kaufmann Publishers.

[BD97a] S. Baluja and S. Davies. Combining multiple optimization runs with optimal dependency trees. Technical Report CMU-CS-97-157, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1997.

[BD97b] S. Baluja and S. Davies. Using optimal dependency-trees for combinational optimization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 30–38, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[BDF$^+$03] K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D.M. Blei, and M.I. Jordan. Matching words and pictures. *J. Mach. Learn. Res.*, 3:1107–1135, 2003.

[BHT89] A. Buja, T. Haste, and R. Tibshirani. Linear smoothersand additive models. *The Annals of Statistics*, 17:453–555, 1989.

[Bin81] T.O. Binford. Inferring surfaces from images. In *Artificial Intelligence*, volume 17, pages 205–244, 1981.

[BJ85] P.J. Besl and R.C. Jain. Tree-dimentional object recognition. In *Computing surveys*, volume 17, pages 75–145, 1985.

[BJH$^+$96] J. Bala, K. De Jong, J. Huang, H. Vafaie, and H. Wechsler. Using learning to facilitate the evolution of features for recognizing visual concepts. *Evolutionary Computation*, 4(3):297–311, 1996.

[BL01] R. Blanco and J.A. Lozano. *An empirical comparison of discrete estimation of distribution algorithms*, chapter chapter 7, pages 167–180. Kluwer Academic Publishers, 2001.

[Bla00] S. Blackmore. *The Meme Machine (Popular Science)*. Oxford University Press, USA, May 2000.

[BNJ03] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[Bre96] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[Bre01] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[BS03] B.J. Balas and P. Sinha. Dissociated dipoles: Image representation via non-local comparisons. Annual meeting of the Vision Sciences Society, Sarasota, FL., 2003.

[BS06] B.J. Balas and P. Sinha. Receptive field structures for recognition. *Neural Computation*, 18(3):497–520, 2006.

[BTG06] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. pages 404–417. 2006.

[BU01] E. Borenstein and S. Ullman. Class specific top down-segmentation. *Proceedings of the European Conference on Computer Vision*, pages 110–122, 2001.

[BZ04] N. Barnes and A. Zelinsky. Real-time radial symmetry for speed sign detection. *Proc IEEE Intelligent Vehicles Symposium*, 2004.

[BZM08] A. Bosch, A. Zisserman, and X. Muñoz. Scene classification using a hybrid generative/discriminative approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(4):712–727, April 2008.

[Cal87] W.H. Calvin. Visual features of intermediate complexity and their use in classification. *Nature*, (330):33–34, November 1987.

[Cal97] W.H. Calvin. The six essentials? minimal requirements for the darwinian bootstrapping of quality, May 1997.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.

[CBC] MIT-CBCL face database.

[CEG76] A. Croiser, D. Esteban, and C. Galand. Perfect channel splitting by use of interpolation/decimation/tree decomposition techniques. *Proceedings of the Int. Symp. Info., Circuits, Systems*, 1976.

[CH92] G.F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Mach. Learn.*, 9(4):309–347, 1992.

[Che96] K. Cherkauer. Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks, 1996.

[CJ03a] G. Carneiro and A.D. Jepson. Multi-scale phase-based local features. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 1, June 2003.

[CJ03b] Ud.K. Chakraborty and C.Z. Janikow. An analysis of gray versus binary encoding in genetic search. *Inf. Sci.*, 156(3-4):253–269, 2003.

[CL68] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462–467, May 1968.

[Cor96] Corel stock photo library. [CD-ROM], 1996.

[Cro73] J.L. Crosby. *Computer Simulation in Genetics.* Wiley, New York, 1973.

[CWF76] R. Crochiere, S. Weber, and J. Flanagan. Digital coding of speech in sub-bands. *Bell System Technical Journal*, 55:1069–1085, October 1976.

[DB95] T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[DB05] T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 2005.

[dBCIV97] J.S. de Bonet, Jr. C.L. Isbell, and P. Viola. MIMIC: Finding optima by estimating probability densities. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 424. The MIT Press, 1997.

[DBS05] F. Dadgostar, A.L.C. Barczak, and A. Sarrafzadeh. A color hand gesture database for evaluating and improving algorithms on hand gesture and posture recognition. *Research Letters in the Information and Mathematical Sciences*, 7:127–134, 2005.

[DC99] M.N. Dailey and G.W. Cottrell. Pca = gabor for expression recognition. Technical report, La Jolla, CA, USA, 1999.

[DeJ75] K.A. DeJong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, Ann Arbor, MI, USA, 1975.

[Dem06] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *JMLR*, 7, January 2006.

[Der87] R. Deriche. Using canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, 1(6):167–187, 1987.

[DFS00] S. Dudoit, J. Fridlyand, and T.P. Speed. Comparition of discrimination methods for the classification of tumors using gene expression data. *Technical Report*, June 2000.

[Die00] T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

[Die02] T.G. Dietterich. *The Handbook of Brain Theory and Neural Networks.* Ed. M.A. Arbib, Cambridge, MA: The MIT Press, 2002.

[DK95] T. Dietterich and E. Kong. Error-correcting output coding corrects bias and variance. *S. Prieditis and S. Russell*, 1995.

[dlEAPR04] A. de la Escalera, J.M. Armingol, J.M Pastor, and F.J. Rodriguez. Visual sign information extraction and identification by deformable models for intelligent vehicles. *Intelligent Transportation Systems, IEEE Transactions on*, 5(2):57–68, June 2004.

[dlEAS01] A. de la Escalera, J.M. Armingol, and M.A. Salichs. Recognition of traffic signs using a multilayer neural network. In *3rd International Conference on Field and Service Robotics*, Espoo, Finland, June 2001.

[DP06] Disney Enterprises inc and PIXAR animation studios. Cars. `http://www.pixar.com/featurefilms/cars/`, 2006.

[DRdR02] R.P.W. Duin, F. Roli, and D. de Ridder. A note on core research issues for statistical pattern recognition. *Pattern Recogn. Lett.*, 23(4):493–499, 2002.

[dSB92] M. de Saint Blancard. Road sign recognition: A study of vision-based decision making for road enviroment recognition. In *Vision–based Vehicle Guidance*, Springer Series in Perception Engineering, pages 162–172, New York, Berlin, Heidelberg, 1992. Springer Verlag.

[Dun61] O.J. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association.*, 56(293):52–64, Mar. 1961.

[DWF$^+$04] C. Dance, J. Willamowski, L. Fan, C. Bray, and G. Csurka. Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*, 2004.

[EL99] R. Etxeberria and P. Larrañaga. Global optimization with bayesian networks. In *II Symposium on Artificial Intelligence. CIMAF99. Special Session on Distributions and Evolutionary Optimization, Cuba.*, pages 332–339, 1999.

[EMSA97] A. De La Escalera, L.E. Moreno, M.A. Salichs, and J.M. Armingol. Road traffic sign detection and classification. In *IEEE: Transactions on Industrial Electronics*, volume 44, pages 848–859. IEEE, Dec 1997.

[EPR06] S. Escalera, O. Pujol, and P. Radeva. Decoding of ternary error correcting output codes. *CIARP*, 2006.

[ER04] S. Escalera and P. Radeva. Fast greyscale road sign model matching and recognition. *Recent Advances in Artificial Intelligence Research and Development, IOS Press, Amsterdam*, October 2004.

[Esc08] S. Escalera. *Coding and Decoding Design of ECOCs for Multi-class Pattern and Object Recognition.* PhD thesis, Computer Science Department, Universitat Autónoma de Barcelona, 2008.

[FA91] W.T. Freeman and E.H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[FB70] A. Fraser and D. Burnell. *Computer Models in Genetics.* McGraw-Hill, New York, 1970.

[FFFT07] L. Fei-Fei, R. Fergus, and A. Torralba. Recognizing and Learning Object Categories. CVPR 2007 Minneapolis, Short Course, June 17, 2007.

[FFP05] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 524–531, Washington, DC, USA, 2005. IEEE Computer Society.

[FGG$^+$98] U. Franke, D. Gavrila, S. Gorzig, F. Lindner, F. Paetzold, and C. Wohler. Autonomous driving goes downtown. *IEEE Intelligent Systems*, 13(6):40–48, 1998.

[FH65] A.S. Fraser and P.E. Hansche. Simulation of genetic systems. major and minor loci. In *Proc. 11th Int. Congress on Genetics*, volume 3, pages 507–516. Ed. Oxford, 1965.

[FHT00] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2):337–407, 2000.

[FLCS05] M. Fritz, B. Leibe, B. Caputo, and B. Schiele. Integrating representative and discriminant models for object category detection. *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 2:1363–1370, Oct. 2005.

[Fle92] M.M. Fleck. Multiple widths yield reliable finite differences (computer vision). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(4):412–429, 1992.

[Fle04] F. Fleuret. Fast binary feature selection with conditional mutual information. *J. Mach. Learn. Res.*, 5:1531–1555, 2004.

[Fog06] D.B. Fogel. Nils barricelli - artificial life, coevolution, self-adaptation. *Computational Intelligence Magazine, IEEE*, 1(1):41–45, Feb. 2006.

[Fra57] A.S. Fraser. Simulation of genetic systems by automatic digital computers. I. Introduction. *Aust. J. Biol. Sci.*, 10:484–491, 1957.

[Fre95] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[FRKV94] L.M.J. Florack, B.M. Ter Haar Romeny, J.J. Koenderink, and M.A. Viergever. General intensity transformations and differential invariants. *Journal of Mathematical Imaging and Vision*, 4(2):171–187, 1994.

[FS81] J. Friedman and W. Stuetzle. *Projection pursuit regression.* J. Amer. Statist. Assoc. 76 report, Stanford University., 1981.

[FS96] Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[FS97] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[GBS05] J.M. Geusebroek, G.J. Burghouts, and A.W.M. Smeulders. The Amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112, 2005.

[GGL98] A. Guarda, C. Le Gal, and A. Lux. Evolving visual features and detectors. In *SIBGRAPHI '98: Proceedings of the International Symposium on Computer Graphics, Image Processing, and Vision*, pages 246–254, Washington, DC, USA, 1998. IEEE Computer Society.

[GLY94] D. Ghica, S.W. Lu, and X. Yuan. Recognition of traffic signs using a multilayer neural network. In *Proceedings of the 1994 Canadian Conference on Electrical and Computer Engineering*, volume 44, pages 848–859, Halifax, Canada, 1994.

[Gol90] D.E. Goldberg. A note on boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Systems*, (4):445–460, 1990.

[GRS96] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter. *Markov Chain Monto Carlo in Practice.* Interdisciplinary Statistics Series. CRC Press, 1996.

[GWKS07] M. Gallagher, I. Wood, J. Keith, and G. Sofronov. Bayesian inference in estimation of distribution algorithms. *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 127–133, Sept. 2007.

[Har28] R.V. Hartley. *Transmission of information.* Number 7. Bell System Technical Journal, 1928.

[Har99] G. Harik. Linkage learning via probabilistic modeling in the ecga. Technical Report 99010, University of Illinois at Urbana-Champaign, 1999.

[HCL02] C. Hsu, C. Chang, and C. Lin. A practical guide to support vector classification. *Department of CSIE, technical report*, 2002.

[Hen88] M. Henrion. Propagating uncertainty in bayesian networks by probabilistic logic sampling. In *Uncertainty in Artificial Intelligence 2*, pages 149–163. North-Holland, 1988.

[HGC95] D. Heckerman, D. Geiger, and D.M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. volume 20, pages 197–243, September 1995.

[HH01] S. Hsu and C. Huang. Road sign detection and recognition using matching pursuit method. In *Image and Vision Computing*, volume 19, pages 119–129, 2001.

[HKT$^+$98] U. Handmann, T. Kalinke, C. Tzomakas, M. Werner, and W. von Seelen. An image processing system for driver assistance. In *IV'98, IEEE International Conf. on Intelligent Vehicles*, pages 481–486, Stuttgart, Germany, 1998. IEEE.

[HL05] J. Huang and C.X. Ling. Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):299–310, 2005.

[HLG99] G.R. Harik, F.G. Lobo, and D.E. Goldberg. The compact genetic algorithm. *IEEE-EC*, 3(4):287, November 1999.

[Ho98] T.K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[Hof99] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, New York, NY, USA, 1999. ACM.

[Hol75] J.H. Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* University of Michigan Press, 1975.

[HR98] T. Hastie and R.Tibshirani. Classification by pairwise grouping. In *proc. NIPS*, volume 26, pages 451–471, 1998.

[HRB99] D. Howard, S.C. Roberts, and R. Brankin. Evolution of ship detectors for satellite sar imagery. In *EuroGP*, pages 135–148, 1999.

[HT01] D.J. Hand and R.J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Mach. Learn.*, 45(2):171–186, November 2001.

[HW59] D.H. Hubel and T.N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *Journal of Physiology*, (148):574–591, 1959.

[HWH05] A.B. Hillel, D. Weinshall, and T. Hertz. Efficient learning of relational object class models. *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 2:1762–1769 Vol. 2, Oct. 2005.

[ID80] R.L. Iman and J.M. Davenport. Approximations of the critical region of the friedman statistic. In *Communications in Statistics*, pages 571–595, 1980.

[JC82] A.K. Jain and B. Chandrasekaran. Dimensionality and sample size considerations. *Pattern Recognition Practice*, 2(39):835–855, 1982.

[JH99] T.S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 487–493, Cambridge, MA, USA, 1999. MIT Press.

[JP87] J.P. Jones and L.A. Palmer. An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex. *Journal of Neurophysiology*, (58):1233–1258, 1987.

[Kap94] J.N. Kapur. *Measures of Information and Their Applications.* John Wiley & Sons, 1994.

[KB07] K. Krawiec and B. Bhanu. Visual learning by evolutionary and coevolutionary feature synthesis. *IEEE Transactions on Evolutionary Computation*, 11(5):635–650, October 2007.

[KHZ00] Y.H. Kim, S.Y. Hahn, and B.T. Zhang. Text filtering by boosting naïve bayes classifiers. *SIGIR Conference on Research and Development*, 2000.

[KV94] M.J. Kearns and U.V. Vazirani. An introduction to computational learning theory. MIT Press, 1994.

[KZ94] D. Kellmeyer and H. Zwahlen. Detection of highway warning signs in natural video images using color image processing and neural networks. In *Int. Conf. Neural Networks 1994*, volume 7, pages 4226–4231. IEEE, 1994.

[Lar01] P. Larrañaga. *An Introduction to Probailistic Graphical Models*, chapter chapter 2, pages 27–56. Kluwer Academic Publishers, 2001.

[Lau96] S.L. Lauritzen. *Graphical Models.* Oxford University Press, 1996.

[LB03] Y. Lin and B. Bhanu. Learning features for object recognition. In *GECCO*, pages 2227–2239, 2003.

[LD05] D. Lowd and P. Domingos. Naive bayes models for probability estimation. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 529–536, New York, NY, USA, 2005. ACM Press.

[Lev69] M.D. Levine. Feature extraction : A survey. *Proceedings of the IEEE*, 57(8):1391–1407, August 1969.

[LL02] P. Larrañaga and J.A. Lozano, editors. *Estimation of distribution algorithms.* Genetic Algorithms and Evolutionary Computation. Kluwer Academic Publishers, USA, 2002.

[LLIB06] J.A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea. *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms (Studies in Fuzziness and Soft Computing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[LLM03] P. Larrañaga, J.A. Lozano, and H. Mühlenbein. Algoritmos de estimación de distribuciones en problemas de optimización combinatoria. *Inteligencia Artificial, Revista Iberoamericana de IA*, 7(19):149–168, 2003.

[LM02] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Proceedings of the International Conference on Image Processing*, pages 900–903, Rochester, USA, September 2002. IEEE.

[Low99] D.G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the International Conference on Computer Vision ICCV, Corfu*, pages 1150–1157, 1999.

[Low04] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[LSH06] F.G. Lobo, K. Sastry, and G.R. Harik. Extended compact genetic algorithm in c++: Version 1.1. Illinois Genetic Algorithms Laboratory, Report No. 2006012, March 2006.

[LZ03] G. Loy and A. Zelinsky. Fast radial symmetry for detecting points of interest. *IEEE Trans Pattern Analysis and Machine Intelligence*, 25(8), 2003.

[Mar80] S. Marĉelja. Mathematical description of the responses of simple cortical cells. *Journal of Optical Society of America*, 70(11):1297–1300, 1980.

[Mar82] D. Marr. Vision. W.H Freeman and Company, 1982.

[MB98] A.M. Martinez and R. Benavente. The AR Face Database. CVC Technical Report ♯24., June 1998.

[MBL04] T. Miquélez, E. Bengoetxea, and P. Larrañaga. Evolutionary computation based on bayesian classifiers. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *International Journal of Applied Mathematics and Computer Science, 14(3)*, volume 2, pages 101–115, 2004.

[MBLAGJ$^+$07] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno, and F. Lopez-Ferreras. Road-sign detection and recognition based on support vector machines. *Intelligent Transportation Systems, IEEE Transactions on*, 8(2):264–278, June 2007.

[MBS$^+$05] S.R. Madeira, L.C. Bastos, A.M. Sousa, J.F. Sobral, and L.P. Santos. Automatic traffic signs inventory using a mobile mapping system. In *GIS PLANET 2005, International Conference and Exhibition on Geographic Information*, Estoril, Lisboa, Portugal, 2005.

[MH80] D. Marr and E. Hildreth. Theory of edge detection. In *Proceedings of the Royal Society of London. Series B, Biological Sciences*, volume 207, pages 187–217, 1980.

[Mit97] T.M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[MKS00] J. Miura, T. Kanda, and Y. Shirail. An active vision system for real-time traffic sign recognition. In *IEEE: International Conference on Intelligent Transportation System 2000 (ITSC '00)*, pages 52–57, 2000.

[MKSH08] T. Mita, T. Kaneko, B. Stenger, and O. Hori. Discriminative feature co-occurrence selection for object detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(7):1257–1269, July 2008.

[MM99] H. Mühlenbein and T. Mahning. The factorized distribution algorithm for additively decomposed functions. In *Second Symposium on Articial Intelligence. Adaptive Systems, CIMAF 99. La Habana*, pages 301–313, 1999.

[MM01] T. Mahnig and H. Mühlenbein. Comparing the adaptive boltzmann selection schedule sds to truncation selection. In *III Symposium on Artificial Intelligence. CIMAF01. Special Session on Distributions and Evolutionary Optimization.*, pages 121–128, 2001.

[MP88] M. Minsky and S. Papert. Perceptrons. pages 157–169, 1988.

[MS05] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, 2005.

[MSV93] H. Mühlenbein and D. Schlierkamp-Voosen. The science of breeding and its application to the breeder genetic algorithm (bga). *Evol. Comput.*, 1(4):335–360, 1993.

[MTEF06] K. Murphy, A. Torralba, D. Eaton, and W. Freeman. Object detection and localization using local and global features. *Toward Category-Level Object Recognition, Lecture Notes in Computer Science*, 4170:382–400, January 2006.

[MTGM04] F. Mindru, T. Tuytelaars, L. Van Gool, and T. Moons. Moment invariants for recognition under changing viewpoint and illumination. *Comput. Vis. Image Underst.*, 94(1-3):3–27, 2004.

[Müh97] H. Mühlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5(3):303–346, 1997.

[Nem63] P.B. Nemenyi. *Distribution-free multiple comparisons.* PhD thesis, Princeton University, 1963.

[Nil65] N.J. Nilsson. *Learning Machines.* McGraw-Hill, 1965.

[NJ02] A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in Neural Information Processing Systems*, (14):841–848, 2002.

[Par62] E. Parzen. On the estimation of probability density function and the mode. *The Annals of Mathematical Statistics*, page 1065, 1962.

[PD07] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.

[Per08] F. Perronnin. Universal and adapted vocabularies for generic visual categorization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(7):1243–1256, July 2008.

[PFX00] J.C. Principe, J.W. Fisher, and D. Xu. *Information theoretic learning.* Unsupervised Adaptive Filtering. Simon Kaykin, editor, New York, 2000.

[PGCP99] M. Pelikan, D.E. Goldberg, and E. Cantú-Paz. BOA: The Bayesian optimization algorithm. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, volume I, pages 525–532, Orlando, FL, 13-17 1999. Morgan Kaufmann Publishers, San Fransisco, CA.

[Pla83] R.L. Plackett. Karl pearson and the chi-squared test. *International Statistical Review / Revue Internationale de Statistique*, 51(1):59–72, Apr. 1983.

[PM99] M. Pelikan and H. Mühlenbein. The bivariate marginal distribution algorithm. In R. Roy, T. Furuhashi, and P. K. Chawdhry, editors, *Advances in Soft Computing - Engineering Design and Manufacturing*, pages 521–535, London, 1999. Springer-Verlag.

[PMC96] Gi. Piccioli, E. De Micheli, and M. Campani. A robust method for road sign detection and recognition. In *ECCV (1)*, pages 495–500, 1996.

[PMPC94] G. Piccioli, E.D. Michelli, P. Parodi, and M. Campani. Robust road sign detection and recognition from image sequences. *Proc. Intelligent Vehicles*, pages 278–283, 1994.

[PND06] P. Paclik, J. Novovicova, and R.P.W. Duin. Building road-sign classifiers using a trainable similarity measure. *Intelligent Transportation Systems, IEEE Transactions on*, 7(3):309–321, Sept. 2006.

[PR81] D.A. Pollen and S.F. Ronner. Phase relationship between adjacent simple cells in the visual cortex. *Science*, (212):1409–1411, 1981.

[PRV06] O. Pujol, P. Radeva, and J. Vitrià. Discriminant ECOC: A heuristic method for application dependent design of error correcting output codes. *Transactions on PAMI*, 28(6):1001–1007, 2006.

[Rec71] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* PhD thesis, Technical University of Berlin, Department of Process Engineering, 1971.

[Ren61] A. Renyi. On mesures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, pages 547–561. Universtiy of California Press, 1961.

[Ren76] A. Renyi. Some fundamental questions of information theory. In *Selected Papers of Alfred Renyi*, volume 2, pages 526–552, Budapest, 1976. Akademia Kiado.

[Ris78] J. Rissanen. Modeling by shortest data description. *Automatica*, pages 465–471, 1978.

[San] R. Santana. MatEDA. Intelligent System Group, Department of Computer Science and Artificial Intelligence, University of the Basque Country.

[SBC08] J. Shotton, A. Blake, and R. Cipolla. Multiscale categorical object recognition using contour fragments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(7):1270–1281, July 2008.

[Sch78] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[Sch81] H.P. Schwefel. *Numerical optimization of computer models.* Wiley, New York, 1981.

[Sch87] J.D. Schaffer. Some effects of selection procedures on hyperplane sampling by genetic algorithms. In *In, Genetic Algorithms and Simulated Annealing*, pages 89–130. Pitman, 1987.

[Sch90] R.E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[SGS00] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search.* The MIT Press, 2000.

[She00] D.J. Sheskin. *Handbook of parametric and nonparametric statistical procedures.* Chapman & Hall/CRC, 2000.

[Sil86] B.W. Silverman. *Measures of Information and Their Applications.* Chapman and Hall, London, 1986.

[SKJ99] M. Schmidt, K. Kristensen, and T.R. Jensen. Adding genetics to the standard PBIL algorithm. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 2, pages 1527–1534, 1999.

[SLDV98] P. Simard, Y. LeCum, J. Denker, and B. Victorri. Transformation invariance in pattern recognition, tangent distance and tangent propagation. *Neural Networks: Tricks of the Trade*, 1524:239–274, 1998.

[SOP07] K. Sastry and A. Orriols-Puig. Extended compact genetic algorithm in matlab. Technical Report No. 2007009. Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 2007.

[SPG$^+$02] D.G. Shaposhnikov, L.N. Podladchikova, A.V. Golovan, N.A. Shevtsova, K. Hong, and X.W. Gao. Road sign recognition by single positioning of space-variant sensor, 2002.

[SR07] S. Sedai and P.K. Rhee. Bio-inspired adaboost method for efficient face recognition. *Frontiers in the Convergence of Bioscience and Information Technologies, 2007. FBIT 2007*, pages 715–718, Oct. 2007.

[SRE$^+$05] J. Sivic, B.C. Russell, A.A. Efros, A. Zisserman, and W.T. Freeman. Discovering objects and their localization in images. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pages 370–377, Washington, DC, USA, 2005. IEEE Computer Society.

[SSCG06] Y. Su, S. Shan, X. Chen, and W. Gao. Patch-based gabor fisher classifier for face recognition. *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, 2:528–531, 0-0 2006.

[SZ02] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "how do i organize my holiday snaps?". In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I*, pages 414–431, London, UK, 2002. Springer-Verlag.

[TC00] K. Torkkola and W. Campbell. Mutual information in learning feature transformations. In *Proceedings of ICML 2000*, 2000.

[TM04] A. Torralba and K. Murphy. Sharing visual features for multiclass and multiview object detection. In *TPAMI*, 2004.

[TMF07] A. Torralba, K.P. Murphy, and W.T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):854–869, 2007.

[TS01] K. Thoresz and P. Sinha. Qualitative representations for recognition. *Journal of Vision*, 1(3):298–298, 12 2001.

[TZ04] A. Treptow and A. Zell. Combining adaboost learning and evolutionary search to select features for real-time object detection. *Evolutionary Computation, 2004. CEC2004. Congress on*, 2:2107–2113, June 2004.

[Ull96] S. Ullman. High-level vision. object recognition and visual cognition. In *A Bradford Book*, New York, 1996. The MIT Press.

[US00] S. Ullman and E. Sali. Object classification using a fragment-based representation. In *BMVC '00: Proceedings of the First IEEE International Workshop on Biologically Motivated Computer Vision*, pages 73–87, London, UK, 2000. Springer-Verlag.

[USVN01] S. Ullman, E. Sali, and M. Vidal-Naquet. A fragment-based approach to object representation and classification. *International Workshop on Visual Form, Berlin: Springer*, pages 85–100, 2001.

[UVNS02] S. Ullman, M. Vidal-Naquet, and E. Sali. Visual features of intermediate complexity and their use in classification. *Nature Neuroscience*, 7(5):1–6, 2002.

[vdWB06] C. van der Walt and E. Barnard. Data characteristics that determine classifier performance. *17th Annual Symposium of the Pattern Recognition Association of South Africa*, December 2006.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–511–I–518, 2001.

[VL89] M. Vetterli and D. LeGall. Perfect reconstruction FIR filter banks: some properties and factorizations. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(7):1057–1071, 1989.

[VNU03] M. Vidal-Naquet and S. Ullman. Object recognition with informative features and linear classification. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 281, Washington, DC, USA, 2003. IEEE Computer Society.

[VS04] J. Vogel and B. Schiele. Natural scene retrieval based on a semantic modeling step. *Image and Video Retrieval*, 3115:207–215, 2004.

[VV88] R.L. De Valois and K.K. De Valois. *Spatial Vision.* Oxford Press, 1988.

[Whi89] L.D. Whitley. The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 116–123, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[Whi94] D. Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4(2):65–85, June 1994.

[WP99] R.A. Watson and J.B. Pollack. Hierarchically consistent test problems for genetic algorithms. *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, 2:1406–1413, 1999.

[YFW] J.S. Yedidia, W.T. Freeman, and Y. Weiss. Generalized belief propagation. In *NIPS*, pages 689–695.

[Zar98] J.H. Zar. *Biostatistical Analysis.* Prentice Hall, 1998.

Publications

Conferences

X. Baró and J. Vitrià, "Fast Traffic Sign Detection on greyscale images". 7th Congrés Català d'Intel·ligència Artificial, Barcelona. In press: Recent Advances in Artificial Intelligence Research and Development, Frontiers in Artificial Intelligence and Applications, $113 : 209 - 216$. IOS Press, ISBN: $978 - 1 - 58603 - 466 - 5$. October 2004.

X. Baró and J. Vitrià, "Feature Selection with Non-Parametric Mutual Information for Adaboost Learning". 8th Congrés Català d'Intel·ligència Artificial, Alguero, Italy. In press: Artificial Intelligence Research and Development, Frontiers in Artificial Intelligence and Applications, $131 : 131 - 138$. IOS Press, ISBN: $1 - 58603 - 560 - 6$. October 2005.

X. Baró and J. Vitrià, "Real-Time Object Detection using an Evolutionary Boosting Strategy". 9th Congrés Internacional de l'Associació Catalana d'Intel·ligència Artificial, Perpignan, France. In press: Artificial Intelligence Research and Development, Frontiers in Artificial Intelligence and Applications, $146 : 9 - 18$. IOS Press, ISBN: $978 - 1 - 58603 - 663 - 7$. October 2006.

X. Baró and J. Vitrià, "Evolutionary Boosting Strategy". 1st CVC Workshop, Barcelona. In press: Computer Vision: Process of Research and Development. Proceedings of the 1st CVC Workshop, $59 - 62$. ISBN: $84 - 933652 - 8 - 9$. October 2006.

X. Baró and J. Vitrià, "Weighted Dissociated Dipoles for Evolutive Learning". 10th Congrés Internacional de l'Associació Catalana d'Intel·ligència Artificial, Sant Julià de Loria, Andorra. In press: Artificial Intelligence Research and Development, Frontiers in Artificial Intelligence and Applications, $163 : 189 - 196$. IOS Press, ISBN: $978 - 1 - 58603 - 798 - 7$. October 2007.

X. Baró and J. Vitrià, "Visual features selection using Naïve Bayes Models". 2nd CVC Workshop, Barcelona. In press: Computer Vision: Process of Research and Development. Proceedings of the 1st CVC Workshop, $54 - 59$. ISBN: $978 - 84 - 935251 - 4 - 9$. October 2007.

X. Baró and J. Vitrià, "Evolutionary Object Detection by means of Naïve Bayes Models Estimation". 10th European Workshop on Evolutionary Computation in Image Analysis and Signal Processing (EvoIASP 2008), Naples, Italy. In press: Applications of Evolutionary Computing, EvoWorkshops 2008: EvoCOMNET, EvoFIN, EvoHOT, EvoIASP, EvoMUSART, EvoNUM, EvoSTOC, and Evo-TransLog, Naples, Italy, March $26-28$, 2008. Lecture Notes in Computer Science, $4974 : 235-244$. Springer. ISBN: $978-3-540-78760-0$. March 2008.

X. Baró and J. Vitrià, "Weighted Dissociated Dipoles: An extended visual feature set". 6th International Conference on Computer Vision Systems (ICVS 2008), Santorini, Greece. In press: Theoretical Computer Science and General Issues 2008. Lecture Notes in Computer Science, $5008 : 281-290$. Springer. ISBN: $978-3-540-79546-9$. May 2008.

Journals

X. Baró, S. Escalera, J. Vitrià, O. Pujol, and P. Radeva "Traffic Sign Recognition using Evolutionary Adaboost detection and Forest-ECOC classification.". IEEE Transactions on Intelligent Transportation Systems. To be published.

Technical Reports

X. Baró, S. Escalera, P. Radeva, and J. Vitrià "Informe Final Detecció i Classificació de Senyals de Trànsit", CVC-ICC 052005, CVC (UAB). May 2005.

X. Baró, "Fast traffic sign detection on gray-scale images", CVC Technical Report #82, CVC (UAB). July 2005.

X. Baró, S. Escalera, P. Radeva, and J. Vitrià "Detecció de regions de Text.", CVC-ICC 062008, CVC (UAB). June 2008.

X. Baró, S. Escalera, P. Radeva, and J. Vitrià "Reconeixement de Guals en entorns urbans", CVC-ICC 122008, CVC (UAB). December 2008.

Progress Report

X. Baró, S. Escalera, P. Radeva, and J. Vitrià "Informe de Planificació Detecció i Classificació de Senyals de Trànsit.", CVC-ICC 112005, CVC (UAB). November 2005.

X. Baró, S. Escalera, P. Radeva, and J. Vitrià ”Reconeixement de senyals de trànsit. Línies de continuació.”, CVC-ICC 062006, CVC (UAB). June 2006.

X. Baró, S. Escalera, P. Radeva, and J. Vitrià ”Reconeixement de senyals de trànsit. Detecció de Guals.”, CVC-ICC 112006, CVC (UAB). November 2006.

X. Baró, S. Escalera, P. Radeva, and J. Vitrià ”Detecció de regions de Text.”, CVC-ICC 122007, CVC (UAB). December 2007.

Awards

Nomination to the best paper award of the 10^{th} European Workshop on Evolutionary Computation in Image Analysis and Signal Processing (EvoIASP 2008)