Università
Ca'Foscari
Venezia

MSc (*ex D.M. 270/2004*)
in Computer Science

Dissertation

–

Ca' Foscari
Dorsoduro 3246
30123 Venezia

A Game-Theoretic Approach to Graph Transduction:
An Experimental Study

Supervisor
Prof. Marcello Pelillo

Candidate
Michele Schiavinato
Id 810469

Academic Year
2012 / 2013

CA' FOSCARI UNIVERSITY - VENICE

DEPARTMENT OF ENVIRONMENTAL SCIENCES,
INFORMATICS AND STATISTICS
Second Cycle Degree Programme in Computer Science

Dissertation

A Game-Theoretic Approach to Graph Transduction:

An Experimental Study

Student:
Michele Schiavinato
SID 810469

Supervisor:
Prof. Marcello Pelillo

Academic Year 2012-2013

Contents

Abstract

An important class of problems in machine learning is based on semi-supervised process to look for consistent labeling for a set of objects. The crucial aspect which features this topic is how to spread the available knowledge on the data to infer a proper answer for the unknown one. In this dissertation we use an important graph based model of semi-supervised learning known as Graph Transduction. Our research begins from a novel solution in Game Theory which models the problem just a noncooperative game: the Graph Transduction Game (GTG). We study this algorithm for the specific instance of visual world, leading the analysis over the main information of visual similarity between images and measures of similarity projected to categories. Finally we introduce the implications arisen on large-scale classification, guessing a possible solution generalizing the current version of GTG, which can exploit of category similarities actively in the learning.

Document Notation

Multidimensional objects

- a: is any generic element (normally can assume nominal, scalar or vectorial form);
- $\mathbf{a} = (a_1, a_2, \ldots, a_m)^T$: is a column vector of m elements (or a $m \times 1$ matrix, with m rows and one column);
- $\mathbf{A} = (a_{ij})$: is a $m \times n$ matrix, with m rows and n columns where a_{ij} is the element in the matrix at the row i and column j (if a_{ij} is even a matrix the structure $\mathbf{A}$ is said *block matrix*);
- $\mathbf{0}$: is a constant column vector or matrix whose all elements are 0 (the dimensions are explicated by the context);
- $\mathbf{I}_m$: is an identity $m \times m$ matrix composed by all zero values and ones in the main diagonal;
- $\mathbf{e}^h \in \{0, 1\}^m$: is a binary vector of m dimensions where the unique value equals to 1 is the h-th component.

Sets

- $S = \{s_1, s_2, \ldots, s_m\}$: is a finite set composed by a homogeneous collection of m different objects ($|S| = m$ denotes the *cardinality* of the set as the number of its elements), if $m = 0$ then S is expressed as the empty set $\emptyset = \{\}$);
- $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$: are the typical infinite numerical sets respectively for natural, integer, rational, real and complex numbers (to denote the subset of all the positive or negative numbers is added the superscript $+$ or $-$);
- $[\alpha, \beta] \subset \mathbb{R}$: is the *closed* interval for all the real numbers between two scalar ends with $\alpha \leq \beta$ (to denote that α is not included in the interval the symbol '[' is replaced with '(' and in the same manner for β where ']' becomes ')', in this case the interval is said *open* in one on both ends and infinite $\pm\infty$ ends are possible);

- $\mathscr{P}(S)$: is the *power set* of any set S which contains all the possible subsets of S including the empty set $\emptyset$ and itself.

- $\Delta_m = \{\mathbf{x} \in \mathbb{R}^m \mid \forall i = 1 \dots m : x_i \geq 0, \sum_{i=1}^m x_i = 1\}$: is the *Standard simplex* set of m variables and contains any probability distributions in $\mathbb{R}^m$ (let $\mathbf{x} \in \Delta_m$, if $\mathbf{x} = \mathbf{e}^h$ for some $h = 1 \dots m$ then $\mathbf{x}$ is a *vertex point* in the boundary of the standard simplex, otherwise is any other point in the interior space).

- $\sigma(\mathbf{x}) = \Big\{i \in \{1, 2, \dots, m\} \mid x_i \neq 0\Big\}$: is the *support* of a scalar vector $\mathbf{x}$ of m elements as the set of all the indexes where are placed its not null components.

Operations

- $S \times Q = \{(s, q) \mid s \in S, q \in Q\}$: the *Cartesian product* between the sets S and Q is a non commutative operation which returns a set composed by all the possible $|S||Q|$ ordered pairs from the elements contained in them;

- $S^n = \underbrace{S \times S \times \dots \times S}_{n \text{ times}}$: is the *Cartesian power* of a set S as the Cartesian product of itself for n times;

- $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ (with $n, k \in \mathbb{N}; 0 \leq k \leq n$): the *binomial coefficient* which determines the number of subsets of k samples without repetitions extracted from a population of n different elements;

- $\exp(x) = e^x$: is a compact notation for a scalar exponentiation with the typical Euler's number e;

- $(\mathbf{a})_k = a_k$: is a shortcut notation to denote the k-th elements extracted from a vector $\mathbf{a}$;

- $(\mathbf{A})_k = \mathbf{a}_k$: is a shortcut notation to denote the k-th column vector extracted from a matrix $\mathbf{A}$;

- $\|\mathbf{a}\| = \sqrt{\sum_{k=1}^m a_k^2}$: is the *Euclidean norm* (or *length* in the Euclidean space) of a scalar vector of m dimensions;

- $\|\mathbf{a} - \mathbf{b}\| = \sqrt{\sum_{k=1}^m \Big(a_k - b_k\Big)^2}$: is the *Euclidean distance* between two scalar vectors of both m dimensions (you can observe that the norm of a vector is equal to its distance with respect to the origin, namely $\|\mathbf{a}\| = \|\mathbf{a} - \mathbf{0}\|$;

- $\mathbf{a} \cdot \mathbf{b} = \sum_{k=1}^{m} a_k b_k = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos(\theta)$: is the *dot product* (also known as *scalar product* or *inner product* in Euclidean space) ot two scalar vectors where θ denotes the angle between them;
- $\mathbf{A}^T$: the typical *transposition* for a matrix $\mathbf{A} = (a_{ij})$ where $\mathbf{A}^T = (a_{ji})$;
- $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$: given the $m \times n$ matrix $\mathbf{A}$ and the $p \times q$ matrix $\mathbf{B}$ the *Kronecker product* between them returns $\mathbf{C} = (\mathbf{C}_{ij})$ as a $m \times n$ block matrix (or an extended $mp \times nq$ matrix) where the element at row i and column j is a $p \times q$ matrix such that $\mathbf{C}_{ij} = a_{ij}\mathbf{B}$.

Any exception in this dissertation of the notation as above is locally explained to avoid misunderstanding.

Chapter 1

Introduction

In machine learning community the most part of problems to deal with generally is mainly ascribable to a categorization task, which consists in the estimation of a suitable labeling assignment for a given set of objects. Although the problem may seem easy to define in literature there exists a wide collection of operational approaches for labeling problems, whose differences depend mainly from the nature of the objects of interest, the model which organizes the knowledge of the data and the ways to combine different degrees of information in a computational process which researches the better solution.

Semi-supervised learning (SSL) is a notorious subclass of machine learning problems whose fundamental peculiarity is the employment of both labeled and unlabeled objects in the training phase. A well-known approach to SSL is the *Graph Transduction* problem, which consists in a graph-based data model where objects are mapped to nodes and the weighed edges reflect the similarities among them. This method is modeled over an important property on the data said *cluster assumption*, where the features of the objects tends to form separated groups and two instances in the same cluster should share the same category. According this hypothesis the general principle to solve graph transduction problem is exploiting of the supervised information, the labeled nodes, propagating that toward the unlabeled nodes in order to estimate the consistent labeling for all the objects.

Game Theory [1] is a famous area of the science that models a problem under the abstract depiction of game, wherein players act to maximize their score. In this dissertation we deal with a graph transduction problem beginning from a novel solution already existing in the machine learning panorama, which is the non-cooperative game known as Graph Transduction Game (GTG) [2]. The solution of this schema is the condition of Nash equilibrium [3], which is the state where all players have performed the best

move simultaneously, that leads to the best labeling for all objects. Moreover GTG can be placed in one-correspondence with another famous problem well-known as Relaxation Labeling (RL) [4], since the Nash equilibrium in GTG matches with the consistency property of the weighted labeling assignments in RL.

In this dissertation we introduce a specific case of study of the Graph Transduction Game where objects to classify are visual images. For this reason we focus our work treating typical topics of Computer Vision area (a sub-discipline of machine learning) to deal with an image recognition problem in semi-supervised learning. In general terms all our analysis is governed considering two fundamental types of knowledge over the data:

- *visual similarity* as a measure to evaluate how much two images are visually similar;
- *category similarity* as a measure to evaluate how much two classes are similar according information either of visual nature or as high level semantic relations from language ontology.

We are interested to reason about the relationships between visual and semantic category similarity to understand how GTG is affected. Our survey begins to consider the general case wherein the learning exploits of visual similarity only. In this setting we study the properties of the input data and the consequences through a detailed evaluation of the results. In particular we are interested to outline the fundamental aspects that involve the learning, with the fundamental goal to understand the strict bound between the similarity among objects and categories with respect to the classification performances. In this survey we introduce the challenging task of large-scale classification and the implications which takes in semi-supervised learning. In that we finally attempt to guess an experimental form of learning as a generalization of the original GTG algorithm, wherein similarity of categories is effectively used to categorize data: the *Graph Transduction Game with Category Similarity* (GTGwCS). Our experimental solution is studied in detail with the aim to observe what occurs when GTG is released from its original formulation, which is heavy rooted on cluster hypothesis.

The organization of this dissertation is as follows.

- **Chapter 2**. The second chapter introduces some common topics regarding the machine learning field particularly followed to perform this work. In detail it begins from the basis, describing the typical data model whose normal datasets are designed and the fundamental kinds of learning approaches related to them (with major interest toward semi-supervised learning), analyzing dynamical aspects of the possible

computational tasks too. Moreover there are collected some important measures of similarity/dissimilarity over the domains of the training objects or nominal categories. Then the chapters covers some suggestions of proper data pre/post processing to prepare or improve the learning process. Finally there is presented a set of methods both to perform data analysis of the training data and to evaluate the performances given by an estimated model.

- **Chapter 3**. The third chapter presents the fundamental basis of Game Theory science, concentrating on a particular class of games built over certain assumptions and defined in a general formal schema: the *normal-form game* [3]. In particular it explains the properties requested to find a solution of a multiplayer game and a general dynamical approach to compute it. Finally there are introduced some game categories expressed in a *succinct formulation* focusing in deeper the class of *Polymatrix games*, which is indeed essential in this dissertation.

- **Chapter 4**. The fourth chapter covers the definition of the well-known *Labeling problem*. The explanation begins in general terms introducing formally the parallel problem said *Constraint Satisfaction Problem* (CSP) [5, 6], which gives the basis about the concept of consistent labeling assignment. CSP is necessary to introduce its famous generalization, which is a stochastic model definition whose solution is computed by the *Relaxation Labeling process*. Moreover there are other topics covered in detail about the property of consistency for the labeling assignments through relaxation labeling and the dynamical aspects involved in this particular process. The chapter finishes showing an important correspondence of the Relaxation Labeling model with the polymatrix game in Game Theory.

- **Chapter 5**. The fifth chapter introduces a class of problem in semi-supervised learning well-known as *Graph Transduction*. First of all it presents the essential definition of the graph-model inferred from training data and the general goal of the problem. Then the chapter begins its main topic introducing the basis of the transductive learning followed to design a novel solution said *Graph Transduction Game* (GTG), which is a reformulation of graph transduction problem as a non-cooperative polymatrix game. GTG is explained in detail showing its formal definition and some important operational aspects regarding the data setting and execution of the concrete computational algorithm. Finally the chapter introduces an experimental variant of GTG, the *Graph Transduction Game with Category Similarity* (GTGwCS), which employs of an additional information in the learning process about the affinity among categories (from data objects or in semantic terms), showing formally the basis of such guessed schema.

- **Chapter 6**. The seventh chapter summarized all the experimental aspects and results that describe the experience of this work. The first three topics regard initialization tasks before the real execution of GTG and several validation criteria. In particular it introduces all the sources of coarse data explaining those techniques and parameters are set to build the effective datasets employed in the learning processes; then it enters in deep about the construction of the training graph inferred from an input set of data, showing the specific choices and properties followed to obtain a suitable structure; finally there is introduced the main evaluation strategy which is common for a greater part of the analysis, showing detailed aspects about different forms of performance. Finally the chapter begins to introduce the concrete experimental study of Graph Transduction Game, describing in deep several surveys to understand the behavior of this solution according different conditions. In that there are employed different measures of similarity to figure out the relationship between performances and themselves. The last topic consists in the analysis of the variant GTGwCS, to observe what occurs introducing class similarities in the learning.

- **Chapter 7**. The eighth chapter concludes this dissertation making an overview about all the work performed and reasoning about the obtained results.

Chapter 2

Machine Learning

A main area of the science is featured by the demand to learn something of significant given a set of data. This activity is also well-known as *Data mining* [7], which studies and offers useful tools to extract new information from other information, where the latter seems to say nothing of really general. The goal consists to discover non-trivial relationships that can help to predict the possible future behaviors of the objects involved in the context of interest. The request to get automatic these mining methods gives rise of computational solutions, whose fundamental processing aspects are treated in the parallel area of *machine learning* [8].

In this section we introduce basic notions and more specific topics, which are mainly of reference to follow properly our work.

2.1 Machine learning bases

2.1.1 Data model

The preliminary subject focuses about the data model devised to depict the knowledge in a learning process. In this field there exist different ways to formulate a set of data, but the most general employed in practice is based on the concept of *record*. Any real object has measurable features, e.g. for an image could be the pixel color levels or for an animal some physical information (e.g. *height*, *weight*, *age*, *sex*, and so on). In general a single feature k is well defined in a proper domain $\mathcal{F}_k$ which may be numerical/quantitative (e.g. *height*) or nominal/categorical (e.g. *sex*). Therefore considering the representation of any object according a set of q known features, it is possible to define a common descriptor domain for all the objects of interest, which can be modeled as the feature space $\mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2 \times \ldots \times \mathcal{F}_q$. Formally let $\mathcal{B} = \{1, 2, \ldots, n\}$ the collection of n objects and $\mathcal{F}$ the descriptor domain of q features, for each individual $i \in \mathcal{B}$ its *observation* or *feature vector* is the

structure:

$$\mathbf{f}_i = \Big(f_i(1), f_i(2), \ldots, f_i(q)\Big)^T \in \mathcal{F}$$

A good condition which is requested on the data that describes an object is the lack of dependance among its features, namely a feature should not be obtainable from other features (e.g. *age* vs. *birth date*). Moreover typical algorithms are designed to work only with points over the real space, in other terms $\mathcal{F} = \mathbb{R}^q$. Anyway nominal feature can be easily mappable as a quantitative finite domain, therefore we consider implied such preprocessing step.

In a wide class of applications is necessary to introduce an additional important feature associated to the data, whose concrete meaning depends both from the problem of interest and what learning approach is employed (see section 2.1.2). In detail it is sufficient to introduce in deep the idea of *label* or *class*, which limits the domain of this feature to categorical case. We can express formally the space of m possible labels as $\mathcal{Y} = \{1, 2, \ldots, m\}$,[1] therefore for each object $i \in \mathcal{B}$ the data model is extended as the pair

$$(\mathbf{f}_i, y_i) \in \mathcal{F} \times \mathcal{Y}$$

where y_i is the label associated to feature vector $\mathbf{f}_i$ of the i-th object. Such new information y_i is considered as the depended feature with respect to all the others contained in $\mathbf{f}_i$, which is hence an independent component; in other terms the label is assumed to be determinable on the basis of own associated descriptor. This model reflects the well-known *classification problem* in machine learning.[2]

A *data set* may be seen as a non empty finite instance in $\mathscr{P}(\mathcal{F})$ and/or $\mathscr{P}(\mathcal{F} \times \mathcal{Y})$ which collects the information available for all the objects of interest, in general may be arranged as one of the following forms:

1. $\{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n\}$ with the observations only;
2. $\{(\mathbf{f}_1, y_1), (\mathbf{f}_2, y_2), \ldots, (\mathbf{f}_n, y_n)\}$ with the observations and labels;
3. $\{(\mathbf{f}_1, y_1), (\mathbf{f}_2, y_2), \ldots, (\mathbf{f}_k, y_k)\}, \{\mathbf{f}_{k+1}, \mathbf{f}_{k+2}, \ldots, \mathbf{f}_n\}$ with the observations but just a portion is labeled.

[1] In this dissertation for the majority of the cases we consider the domain of categories as a set of numerical indexes, but in other contexts we use the same symbol $\mathcal{Y}$ even if it contains textual labels; although we retain quite easy for the reader to catch the correct meaning. For this reason we avoid the introduction of further verbose notation to distinguish such two aspects.

[2] The case when holds $\mathcal{Y} \subseteq \mathbb{R}$ determines another class of learning which is the field of *regression problem*; anyway it is skipped since not pertinent with respect to the work in this dissertation.

2.1.2 Learning categories

The dataset used in the learning phase, which is generally said *training set*, covers a predominant role since is the unique source of information exploitable to discover relevant patterns, and according the possible forms listed in the section 2.1.3 are associated respectively three principal learning categories.

1. **Unsupervised learning**. In this application is employed a training set without labels: the unique information available is the feature descriptors of the objects. In machine learning there exist different techniques based on this assumption (e.g. Hidden Markov models, Blind Signal Separation), but in this work is important to treat the well-know *Clustering* approach, which exploits of the notions of similarity or distance between objects to group them in ideal sets said *clusters*. The relationship of these groups can be very different, they could be partitions of the data (partitional clustering), or the clusters may share objects (hierarchical clustering) or it is important to discover an unique predominant group (one-class clustering). The classical interpretation of a cluster is the membership to some significant data category (e.g. a set of images of a soccer ball). The main goal followed in this analysis consists to maximize the similarities between objects in the same group (intra-class similarity) and to minimize the similarities between objects from different groups (inter-class similarity) .

2. **Supervised learning**. The training set in this approach contains complete labeled data. The introducing of a fixed domain of labels gives an important contribute to the problem space, because the categories of interest are predetermined and the association between object descriptor and label is known. Typical model based on supervised learning is said *classifier*, which produce a mapper $\Psi : \mathcal{F} \to \mathcal{Y}$ between set of features and response label space according that learned from the training set.

3. **Semi-supervised learning** [9]. This is the particular case where the training set is labeled partially; semi-supervised learning may be seen as an approach that falls between the supervised and unsupervised learning methods. The principal goal is to produce a classifier able to map properly both label and unlabeled data. Clearly to be really useful as technique the ratio of supervised objects is supposed to be much more small with respect to rest of the unsupervised objects. In machine learning the algorithms which deal with semi-supervised learning make at least one of these several assumptions over the training set:

    - *Smoothness*. Objects which are very similar are more likely to share the same label.

- *Cluster*. The global data tends to be organized in different separated clusters and objects in the a same cluster are more likely to share the same label. This property allows that objects which share a same label could be placed also in different cluster (e.g. if a group contains cats and another dogs two elements from different clusters can share the same label *pet*, in other terms an unique label may be organized in multiple separated clusters). Cluster assumption can be seen as a special case of smoothness hypothesis.
- *Manifold*. The labeled and unlabeled information lies approximately on a manifold of much lower dimension than the input space $\mathcal{F}$. This assumption allows to deal with high dimensional data and proceeds to classification using the properties of the estimated manifold.

There exists two fundamental kinds of data inferences which a learning process can be motivated:

- *inductive learning*: reasoning on an observed training set of examples to infer a general rule which is applicable for any other unknown (external) testing examples, i.e. not used to estimated the rule.
- *transductive learning*: reasoning on an observed set of examples from a specific portion of training cases to unknown testing cases to infer a rule which is valid only within that whole set.

In general the definition of semi-supervised learning is sufficiently flexible to support both inductive and transductive inferences, although the latter would be more preferable; as concern supervised learning instead just inductive-based models can be treated.

For any form of learning, the concrete application can be oriented according two main ways:

- **Off-line learning**. The training set used for the learning phase does not change during the time. This possibility is important when it is requested the building of models assumed to remain static for long periods. The condition allows to get acceptable the employing of learning algorithms with a high time complexity for the problem treated.
- **On-line learning**. The training set is a data source which evolves during the time. The behavior of the produced models is dynamical and have to be updated very frequently during the time. Under this condition the time of execution of the learning algorithms becomes relevant, since the new inferred models have to get available with the same speed of the data changing.

2.1.3 Dynamics of the learning

The learning algorithms from a computational point of view may be implemented in different ways and according the problem space have different complexity time order. In general there exist two main family of processes:

- **Statical learning**. The model is inferred follows a determined set of finite steps, the process hence has a polynomial time with respect to the number of objects. The reaching of a final solution usually is always possible with weak data conditions over the training set.
- **Dynamical learning**. The model is inferred after a sequence of indeterminable updating steps, the process has not a polynomial time with respect to the number of object. The reaching of a final solution is not always guaranteed, unless are fixed some data constrains over the training set.

In this specific work there are treated dynamical learning-based methods, whose reference description can be given as the well-known theoretical models of **Dynamical systems** [10]. These systems are designed under the fundamental concept of *state* over a time domain $\mathcal{T}$, which can be continuous or discrete. Formally the state of a system according the independent time $t \in \mathcal{T}$ is modeled as a vectorial function $\mathbf{s} : \mathcal{T} \to \mathcal{S}$ as followings

$$\mathbf{s}^{(t)} = \left(s_1^{(t)}, s_2^{(t)}, \ldots, s_m^{(t)}\right)^T \in \mathcal{S}$$

where $\mathcal{S}$ is the *state space* of m dimensions. In other words $\mathbf{s}^{(t)}$ denotes the equation of motion for the state of the dynamical system.[1]

If $\mathcal{T} = \mathbb{R}$ a *dynamical system* can be described as a non linear system of first order differential equations which is formulated as

$$\dot{\mathbf{s}} = \frac{d}{dt}\big(\mathbf{s}^{(t)}\big) = \chi\big(\mathbf{s}^{(t)}\big) \tag{2.1}$$

where the dot symbol is a shortcut notation to specify derivative with respect the time. The **velocity vector** function $\chi : \mathcal{S} \to \mathcal{S}$ governs the behavior of the dynamical system, which depicts a family of functions for each component as followings

$$\chi\big(\mathbf{s}^{(t)}\big) = \left(\chi_1\big(s_1^{(t)}\big), \chi_2\big(s_2^{(t)}\big), \ldots, \chi_m\big(s_m^{(t)}\big)\right)^T$$

Actually this is the definition of a dynamical system said **autonomous**, since the differentiable map χ does not depend from t; the conversely class is

[1] In literature is usual finding the notation $s(t)$ for the system state: we precise that the meaning is just the same, but we prefer the form $\mathbf{s}^{(t)}$ since is more flexible to explain properly all the topics in this dissertation.

well-known as the **non-autonomous** dynamical system, which is featured by the relation $\chi(t, \mathbf{s}^{(t)})$. Anyway we treat in detail only the autonomous model, whose example is the continuous Replicator Dynamic (3.1).

In the discrete domain of time $\mathcal{T} = \mathbb{N}$ can be designed an operator (or map) $\chi_d : \mathcal{S} \to \mathcal{S}$ giving a formulation of the dynamical system as followings

$$\mathbf{s}^{(t+1)} = \chi_d(\mathbf{s}^{(t)})$$

Often in real applications if the principal definition of the problem is a continuous dynamical system of velocity χ_c is estimated a discretization χ_d which maintains much possible its properties. This description is much more suitable to be treated through the iterative constructs given by the typical environments to design algorithms. Examples of these dynamics are the Rosenfeld-Hummel-Zucker rule (4.3) or the discrete Replicator Dynamic (3.2).

When the dynamic system is satisfied can be interpreted as the reaching of a sort of motionless state $\mathbf{s}^{(t)} = \mathbf{s}_e \in \mathcal{S}$ which is well-known as **equilibrium point**. In a continuous dynamical system the equilibrium state is verified when holds

$$\chi(\mathbf{s}^{(t)}) = \mathbf{0} \tag{2.2}$$

which means the evolution of the system is finished and the point $\mathbf{s}_e$ is a solution for (2.1); after all, since $\chi(\mathbf{s}^{(t)})$ is a derivative with respect to time t, integrating on both sides of (2.2) we just get:

$$\begin{aligned} \int \chi(\mathbf{s}^{(t)})dt &= \int \mathbf{0}dt \\ \mathbf{s}^{(t)} &= const \end{aligned}$$

In discrete dynamical system the reaching of an equilibrium point takes to the following relation

$$\chi_d(\mathbf{s}^{(t)}) = \mathbf{s}^{(t)}$$

namely the system has terminated to produce new solutions at a finite time t.

Normally the concept of equilibrium point in this class of functions is often associated to the mathematical definition of *fixed point*, which is a point in the domain of the function χ which basically maps it in itself.

The behavior of a dynamical system while is getting close to an equilibrium point can follow different forms of *stability*, for example is *stable* if does not be affected by small transformations of the equilibrium point, while *unstable* conversely. These considerations are well explained introducing some notions of the Lyapunov stability, formally an equilibrium point $\mathbf{s}_e \in \mathcal{S}$ can be:

- **stable**, when for each its neighbourhood $U_{\mathbf{s}_e} \in \mathcal{S}$ there exists another neighbourhood $V \subset U_{\mathbf{s}_e}$ wherein any trajectories $\mathbf{s}^{(t)}$ which start from any points in V remains in $U_{\mathbf{s}_e}$ for each time $t > 0$;

- **unstable**, when it is not stable;
- **attractive**, if exists its neighbourhood $U_{\mathbf{s}_e} \in \mathcal{S}$ such that for each trajectory $\mathbf{s}^{(t)}$ which starts from a point in $U_{\mathbf{s}_e}$ holds $\lim_{t \to +\infty} \mathbf{s}^{(t)} = \mathbf{s}_e$;
- is **asymptotically stable**, if is stable and attractive.

All the points generated by a dynamical system paints an image of the state space composed by particular features. An attractor is a property of the state space which can be a point or periodic orbit (cycle) whose the proximity trajectories of the dynamical system are led with the flowing of time. The main kinds of attractors are:

- **punctual attractor**: leads to an equilibrium state since the trajectory which is followed by the system degenerates in a point;
- **cyclic attractor**: leads in an unstable state of the system wherein the trajectory repeats itself perpetually.

The state space is partitioned in different areas which depicts the neighbourhood of each attractor, such regions are said *basins of attraction*. In a certain sense when a system enters in a basin of attraction reaches a special form of *dynamical equilibrium*, since its behavior is not chaotic.

Clearly a stable equilibrium point is a solution much more requested in a dynamical system. The *Lyapunov function* $\varphi_\chi : \mathcal{S} \to \mathbb{R}$ (also well-known as **energy function** for a dynamical system) is a fundamental tool which can be associated to dynamical system χ to proof the stability of an equilibrium state. Formally assuming φ_χ is derivable according the time and always positive $\forall \mathbf{s} \in \mathcal{S}$, $\mathbf{s}_e \in \mathcal{S}$ is an equilibrium point and $\varphi_\chi(\mathbf{s}_e) = 0$, if:

- $\dot{\varphi_\chi}(\mathbf{s}) \leq 0 : \forall \mathbf{s} \in U_{\mathbf{s}_e}$ then $\mathbf{s}_e$ is a stable equilibrium point;
- $\dot{\varphi_\chi}(\mathbf{s}) < 0 : \forall \mathbf{s} \in U_{\mathbf{s}_e}$ then $\mathbf{s}_e$ is a *locally* asymptotically stable equilibrium point;
- $\dot{\varphi_\chi}(\mathbf{s}) < 0 : \forall \mathbf{s} \in \mathcal{S}$ then $\mathbf{s}_e$ is a *globally* asymptotically stable equilibrium point.

In other terms $\mathbf{s}_e$ is a point of local or global minimum of φ_χ. An example of Lyapunov function (although it is given as the symmetric formulation to treat maximum equilibrium points) is the Average Local Consistency (4.4).

2.2 Measures of data similarity

An important problem to treat in machine learning consists to definite a way to measure the *affinity* between couples of data entities, as for example

objects or labels. There are two possible forms or nomenclature to express the same concept from opposite point of views:

$d(\cdot)$ **dissimilarity** or **distance**: measures how much two elements are dissimilar, typically ranges in $[0, +\infty)$;

$s(\cdot)$ **similarity**: measures how much two elements are similar, typically ranges in $[0, 1]$.

Although this is a field indeed wide we just introduce the main topics which are fundamental to understand our work. In that since some concept is applicable both working with measure for objects and labels, we may avoid to lose generality using the term "data entities" to merge both cases; therefore a set of data $D = \{\mathbf{f}_1, \mathbf{f}_2 \ldots\}$ or labels $\mathcal{Y} = \{1, 2, \ldots\}$ may be representing as a more general entity set $\mathcal{E} = \{\varepsilon_1, \varepsilon_2, \ldots\}$, in order to mean $\mathcal{E} = D$ or $\mathcal{E} = \mathcal{Y}$ simultaneously.

2.2.1 Similarity/Distance matrix

In machine learning it is widely employed a general structure well-known as *similarity matrix*,[1] which is defined for a certain measure of similarity $s(\cdot)$ applied to the elements of interest. Formally given a set of z data entities $\mathcal{E}$, it is possible to collect all the pairwise measurements in an unique $z \times z$ real-valued matrix $\mathbf{S} = (s_{\alpha\beta})$, where $s_{\alpha\beta} = s(\varepsilon_\alpha, \varepsilon_\beta)$ for two data entities $\varepsilon_\alpha, \varepsilon_\beta \in \mathcal{E}$. In the same way can be defined the *distance matrix*, if the affinity on $\mathcal{E}$ is a distance measure $d(\cdot)$, formally expressed as a $z \times z$ real-valued matrix $\mathbf{D} = (d_{\alpha\beta})$, where $d_{\alpha\beta} = d(\varepsilon_\alpha, \varepsilon_\beta)$. It is important to observe that although the measure denotes as usual pair of elements, the computation that describes $s(\cdot)$ or $d(\cdot)$ may involve the information on ε_α and ε_β in isolation or additional collective data in $\mathcal{E}$. Working with similarity/distance matrix is a useful expedient for different reasons, for example making data analysis, inferring graph structure, performing symmetrization of normalization of measures, avoiding repetitive computation of same measures and so on.

2.2.2 Object-based similarities

The classical approach to compute similarity between two objects in a dataset $D = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n\}$ is based considering how much they are similar from the values related to their descriptors $\mathbf{f}_i, \mathbf{f}_j \in \mathbb{R}^q$. Anyway there exist other much more extended techniques which can take in account other objects which are bound according some criterion to the two instances, as for example a distance relation.

[1] Often the term similarity matrix is used to denote its typical visual depiction or *heatmap* used in several applications for data analysis (see section 2.4).

2.2.2.1 Euclidean distance

In this measure it is exploited the vectorial concept of *Euclidean distance*, which is defined as followings:

$$d_{euc}(\mathbf{f}_i, \mathbf{f}_j) = \|\mathbf{f}_i - \mathbf{f}_j\| = \sqrt{\sum_{k=1}^{q} \Big(f_i(k) - f_j(k)\Big)^2}$$

This formulation present some properties:

- is a symmetric function since $d_{euc}(\mathbf{f}_i, \mathbf{f}_j) = d_{euc}(\mathbf{f}_j, \mathbf{f}_i)$;
- is always positive since $s_{euc}(\mathbf{f}_i, \mathbf{f}_j) \in [0, +\infty)$;
- when $s_{euc}(\mathbf{f}_i, \mathbf{f}_j) = 0$ (minimum distance) the two objects are equal since holds $\mathbf{f}_i = \mathbf{f}_j$;
- when $s_{euc}(\mathbf{f}_i, \mathbf{f}_j) > 0$ (opened upper limit) the two objects are different since holds $\mathbf{f}_i \neq \mathbf{f}_j$.

Whereas $d_{euc}(\mathbf{f}_i, \mathbf{f}_j)$ is a measure of distance it weights actually the dissimilarity between the two descriptors, therefore how much is low much more the two object are similar. There exist several techniques to turn the Euclidean distance in a much more suitable normalized measure of similarity, but the widely employed as for example the Gaussian kernel (2.3) or the more general rule (2.4).

2.2.2.2 Cosine similarity

In this measure, particularly employed on text-based objects, is used the cosine of the angle between two descriptors as measure of similarity. Recalling the fundamental operation of dot product between two vectors as

$$\mathbf{f}_i \cdot \mathbf{f}_j = \sum_{k=1}^{q} f_i(k) f_j(k) = \|\mathbf{f}_i\| \, \|\mathbf{f}_j\| \cos(\theta)$$

where $\theta \in \mathbb{R}$ is their common orientation angle, the cosine similarity is intuitively the measure

$$s_{cos}(\mathbf{f}_i, \mathbf{f}_j) = \frac{\mathbf{f}_i \cdot \mathbf{f}_j}{\|\mathbf{f}_i\| \, \|\mathbf{f}_j\|} = \cos(\theta)$$

This formulation present some properties:

- is a symmetric function since $s_{cos}(\mathbf{f}_i, \mathbf{f}_j) = s_{cos}(\mathbf{f}_j, \mathbf{f}_i)$;
- is independent with respect to the lengths of the two descriptors since their orientation can not change in relation of them, therefore it can not suggest precisely information to compare the descriptors over their feature values since may occur that holds $\mathbf{f}_i \neq \mathbf{f}_j$ even if $s_{cos}(\mathbf{f}_i, \mathbf{f}_j) = 1$;

- may be both positive and negative since $\cos(\theta) \in [-1, 1]$ for definition;
- when $s_{cos}(\mathbf{f}_i, \mathbf{f}_j) = 1$ (maximum similarity) the two descriptors are parallel, intermediate value $s_{cos}(\mathbf{f}_i, \mathbf{f}_j) \in (0, 1]$ is used to measure levels of similarity;
- when $s_{cos}(\mathbf{f}_i, \mathbf{f}_j) = 0$ (neutral similarity) the two descriptors are orthogonal which denotes independence;
- when $s_{cos}(\mathbf{f}_i, \mathbf{f}_j)$ = -1 (minimum similarity) the two descriptors are opposite, intermediate value $s_{cos}(\mathbf{f}_i, \mathbf{f}_j) \in [-1, 0)$ is used to measure levels of dissimilarity;
- when the two descriptors have components positive, i.e. $\mathbf{f}_i, \mathbf{f}_j \in (\mathbb{R}^+)^q$, cosine similarity is always defined in the domain $[0, 1]$;
- can be converted in a raunchy form of distance $d_{cos}(\mathbf{f}_i, \mathbf{f}_j) \in [0, 2]$ as

$$d_{cos}(\mathbf{f}_i, \mathbf{f}_j) = 1 - s_{cos}(\mathbf{f}_i, \mathbf{f}_j)$$

2.2.2.3 Gaussian k-Nearest-Neighbor method

Given a set of n object descriptors $\{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n\}$ the Gaussian k-Nearest-Neighbor is particular technique to build a graph structure which is described by the weighted adjacency matrix $\mathbf{W} = (w_{ij})$, wherein the weight w_{ij} reflects a similarity measure that depends by the k-neighborhood of the pair of objects i and j.

The similarity weights are based often by Gaussian kernel (see section 2.2.2) over the objects. In that, the kernel width space can be generalized as a function

$$\phi : \mathbb{R} \to \mathscr{P}(\mathbb{R})$$

which returns an ideal finite set $\phi(\delta) = \{\sigma_1, \sigma_2, \ldots, \sigma_w\}$ of w kernel widths generated according an input value δ[1] (the cardinality w depends explicitly from ϕ). The real implementation of ϕ may be very different with respect to the nature of the objects, but in general are produced range of linearly spaced numbers or general discrete interval of real-valued values. To compute Gaussian similarity is necessary to choose a proper kernel width, which can be expressed by the shortcut notation $\sigma_z = \Big(\phi(\delta), z\Big)$ that denotes the z-th kernel width extracted from the specific kernel space $\phi(\delta)$.

The whole weighted adjacency matrix construction is led according the specifications of k, ϕ, z and a distance function $d(\cdot)$. In detail for each object

[1] In machine learning the conceptual function ϕ may require additional parameters, anyway in this dissertation is not necessary to generalize further this tool.

$i \in \mathcal{B}$ can be denoted as $N_i^k \subseteq \mathcal{B} \setminus \{i\}$ the set of the k nearest neighbors objects with respect to i (determined according the $d(\cdot)$ measure). The average distance from such neighbors is expressed as

$$r_i^k = \frac{1}{|N_i^k|} \sum_{j \in N_i^k} d(\mathbf{f}_i, \mathbf{f}_j)$$

The quantity r_i^k is used to compute the kernel width space $\phi(r_i^k)$, wherein is selected the z-th kernel width as

$$\sigma^{(i)} = \Big(\phi(r_i^k), z\Big)$$

Finally the values for the object i in the weighted adjacency matrix $\mathbf{W}$ are determined as following:

$$w_{ij} = \begin{cases} \varrho^{\sigma^{(i)}}\Big(d(\mathbf{f}_i, \mathbf{f}_j)\Big) & j \in N_i^k \\ 0 & \text{otherwise} \end{cases}$$

Therefore the similarity measure w_{ij} between the objects i and j is smooth according an proximity degree of the k-neighborhood, cutting off the externals objects.

Since the weighted adjacency matrix $\mathbf{W}$ contains values of similarity, it is implicitly a similarity matrix too; in fact just letting $\mathbf{S} = \mathbf{W} + \mathbf{I}_n$ to express that in the classical centered and normalized formulation, since $\mathbf{W}$ is also a null-diagonal matrix (i.e. $\forall i \in \mathcal{B} : w_{ii} = 0$) and defined in $[0, 1]$ as consequence of the Gaussian kernel.[1] Anyway this method can not guarantee to obtain symmetric measures; if this condition is a problem for the application context may be necessary to execute some further procedure of symmetrization (see section 2.2.4.2).

2.2.3 Label-based similarities

In (semi)supervised learning the categories may refer either to general identifier for groups of data, as for example a set of numerical indexes $\{1, 2, \ldots\}$, or a higher level information as concrete text words in a given language $\{home, cat, flower, \ldots\}$. In both the cases it is possible that different categories may share common aspects, which may be measured in terms of distance or similarity: for example when two category 1 and 2 are associated to images of different mollusks, or the textual labels *pony* and *horse*. An

[1] In this dissertation we distinguish the symbol $\mathbf{S}$ with respect to $\mathbf{W}$ just to specify that the latter is also a weighted adjacency matrix associated to some graph structure. Although we could call them both generically as similarity matrix, if $\mathbf{W}$ contains similarity weights.

interesting aspect consists to observe that in the former case to infer a measure of similarity it is necessary to know the data entities contained in the groups, while in the latter the information of similarity may be establishment both from the data and according the semantic relationship associated to the specif language (which is indeed opposed with respect to their *syntactical representation*). Category similarities may be used for several tasks, such as external information to perform data analysis, to evaluate classification performances and even in the training phase of a classifier.

In this section we introduce several techniques to measure similarities between data categories, on the basis of about is known about them. In order to avoid misunderstanding we prefer to clarify what we mean with certain expressions.

- **semantic*(-based)* similarity**. Refers to a similarity measure about the linguist semantic (i.e. the associated sense in a given language) between/among words, which is assumed to be obtained according a reference ontology which models semantic relationships of terms.
- **object*(-based)* similarity**. Refers to a similarity measure between/among any form objects, which is assumed to be inferred from the information contained in their data descriptors. When the objects of interest are images the same expression may be turned in "visual similarity".
- **category/class similarity**. Refers to a general similarity measure which is projected between/among the labels used in classification problems to distinguish classes/groups of objects. Therefore, if not specified, the source of information used to infer this measure may be both semantic (when the label is text-based) and object based similarities; when it is necessary to specify this sharp difference, the same expression is turned respectively in "semantic*(-based)* category/class similarity" and "object*(-based)* category/class similarity".

2.2.3.1 Semantic-based category similarities

The words of any language have semantic senses which can be connected to other ones by several linguistic relations; therefore these properties may be represented as an interlinked network of senses. The well-known Word-net taxonomy [11] is a typical example to represent these structure and consists in a huge database of lexical terms in English language where nouns, verbs, adjectives and adverbs are grouped into interconnected sets of cognitive synonyms said *synsets*. The fundamental aspect consists that not only WordNet interconnects words together but also the specific sense (synset) of them. Modeling the semantic property of a term as the membership in a given

sense set it is a solution very expressive, since in a language same words may lay in different synsets; this fact describes the linguistic concept of *word acceptation*, e.g. as "tree may" mean the plant in nature or a hierarchical diagram. Moreover the synsets are connected to other ones from a set of several fundamental semantic relations. The most important is said *super-subordinate relation* or *is-a relation* which models the typical hierarchic bound of the noun terms, or the linguistic property of hypernym and hypnoym. A simplified formal definition of this network can be seen as a forest $\mathcal{W} = (V, E, w)$ where V is the set of senses or synsets, $E \subset V^2$ the set of edges or relations and $w : E \to \mathbb{R}$ a function which define a distance measure given an edge.[1] Moreover WordNet has available another kind of textual information which is well-known in linguistics as *text corpus*, which consists in huge and structured sets of texts used for statistical analysis (for example to check occurrences or validate linguistic rules within a specific language territory). In WordNet the terms in corpora are annotated clearly to be linked with the senses in $\mathcal{W}$ and allow to give a sort of semantic definition (or Semantically Tagged glosses) of the synsets. There exists several text corpus available in the WordeNet (version 3.0), but the must general for the English language are the *American National Corpus* (ANC) and the *British National Corpus* (BNC).

In this section we introduce a set of fundamental semantic similarity measures according the WordNet taxonomy [12, 13]; anyway we remark that such metrics would be applicable over any type of sense networks. Formally, denoting with $\lambda, \mu \in \mathcal{Y}$ two text labels and with $\phi_\lambda, \phi_\mu \in V$ the sense nodes which contain them[2], we introduce these fundamental functions:

- $\text{path} : V \to \mathscr{P}(V)$: the set of senses nodes along the path between the parent of ϕ_λ and its root node;
- $\text{L} : V^2 \to \mathbb{R}$: *length* of the shortest path which connects the senses of two terms (measured in edges or nodes), we assume to have always a global root node ϕ_{root} to guarantee the existence of a path between any pair of nodes;
- $\text{D} : V \to \mathbb{N}$: *taxonomy depth* of a sense ϕ_λ, in other terms the number of nodes in the path between the sense node ϕ_λ and its root node ϕ_{root}, i.e. $\text{D}(\phi_\lambda) = \text{L}(\phi_{root}, \phi_\lambda)$;

[1] In detail WordNet structure would be a fusion of four networks, with respect to the different parts of the speech related to *noun*, *verb*, *adjective* and *adverb*. The fundamental and richly backbone of WordNet is the Noun network, wherein we focus our work since the labels of the objects are always nouns.

[2] Actually to obtain a sense node ϕ_λ the system requires also other metadata, for example a parameter to specify the kind of word λ considered, i.e. noun, verb, adjective or adverb.

- LCS : $V^2 \to V$: *Least Common Subsumer* (LCS) or lowest superordinate (LSO), which is the most specific ancestor sense node between two synsets;
- IC : $V \to \mathbb{R}$: *Information Content* (IC) gives a measure of specificity for the synset, formally let $Pr^{(\mathcal{C})}(\phi_\lambda)$ the probability to encounter an instance of the sense ϕ_λ in the text corpus $\mathcal{C}$, the information content is equal to

$$\mathrm{IC}(\phi_\lambda) = -\log\Big(Pr^{(\mathcal{C})}(\phi_\lambda)\Big)$$

Generalizing the explanation we may image to exploit in practice of a function with form

$$s_t^{\mathcal{W}} : V^2 \to \mathbb{R}$$

which computes a semantic similarity measure of type t between two sense nodes whose definition is the followings.

1. **Semantic Relatedness**. Based on the concept of *semantic distance* [13] in the taxonomy which is defined as

$$s_1^{\mathcal{W}}(\phi_\lambda, \phi_\mu) = \frac{|\mathrm{path}(\phi_\lambda) \cap \mathrm{path}(\phi_\mu)|}{\max(|\mathrm{path}(\phi_\lambda)|, |\mathrm{path}(\phi_\mu)|)}$$

   Although this metric is called "distance" has not be confused as a conventional distance measure (see section 2.2), in fact high values suggest that two words share many senses and hence are considerable quite similar: in other terms is a measure of similarity.

2. **Shortest Path Similarity**. Based on the shortest path length $\mathrm{L}(\phi_\lambda, \phi_\mu)$ that connects the senses ϕ_λ and ϕ_μ in the is-a taxonomy.

3. **Leacock-Chodorow Similarity**. Based as an extension of the shortest path similarity wherein is added the information of the maximum depth of the taxonomy in which the senses occur. Formally the Leacock-Chodorow similarity is defined as

$$s_3^{\mathcal{W}}(\phi_\lambda, \phi_\mu) = -\log\left(\frac{\mathrm{L}(\phi_\lambda, \phi_\mu)}{2\max\limits_{\phi \in V} \mathrm{D}(\phi)}\right)$$

4. **Wu-Palmer Similarity**. Based as a combination between the depth of the least common subsumer $\mathrm{LCS}(\phi_\lambda, \phi_\mu)$ of two senses and the length of their shortest paths. Formally the Wu-Palmer Similarity is defined as

$$s_4^{\mathcal{W}}(\phi_\lambda, \phi_\mu) = \frac{2\mathrm{D}\Big(\mathrm{LCS}(\phi_\lambda,\phi_\mu)\Big)}{\mathrm{L}\Big(\phi_\lambda,\mathrm{LCS}(\phi_\lambda,\phi_\mu)\Big)+\mathrm{L}\Big(\phi_\mu,\mathrm{LCS}(\phi_\lambda,\phi_\mu)\Big)+2\mathrm{D}\Big(\mathrm{LCS}(\phi_\lambda,\phi_\mu)\Big)}$$

5. **Resnik Similarity**. Based on the information content of the least common subsumer, in other terms

$$s_5^{\mathcal{W}}(\phi_\lambda, \phi_\mu) = \mathrm{IC}\Big(\mathrm{LCS}(\phi_\lambda, \phi_\mu)\Big)$$

6. **Jiang-Conrath Similarity**. Based as a combination between the information contents and least common subsumer according the relation

$$s_6^{\mathcal{W}}(\phi_\lambda, \phi_\mu) = \Big(\mathrm{IC}(\phi_\lambda) + \mathrm{IC}(\phi_\mu) - 2\mathrm{IC}\Big(\mathrm{LCS}(\phi_\lambda, \phi_\mu)\Big)\Big)^{-1}$$

7. **Lin Similarity**. Based as a combination between the information contents and least common subsumer according the relation

$$s_7^{\mathcal{W}}(\phi_\lambda, \phi_\mu) = \frac{2\mathrm{IC}\Big(\mathrm{LCS}(\phi_\lambda, \phi_\mu)\Big)}{\mathrm{IC}(\phi_\lambda) + \mathrm{IC}(\phi_\mu)}$$

The semantic similarities introduced are all symmetric measures, anyway in terms of domain of definition there are some differences; in the table 2.1 we summarize all these aspects.

Type	*Name*	*Domain*
1	Semantic Relatedness	$[0, 1]$
2	Shortest Path Similarity	$[0, 1]$
3	Leacock-Chodorow Similarity	$\mathbb{R}^+$
4	Wu-Palmer Similarity	$[0, 1]$
5	Resnik Similarity	$\mathbb{R}^+$
6	Jiang-Conrath Similarity	$[0, 1]$
7	Lin Similarity	$[0, 1]$

Table 2.1: Semantic similarity properties

2.2.3.2 Object-based category similarity

In the scenario of supervised learning the similarities among categorized objects may suggest an estimated measure of category similarity. For example, if the most part of objects from a certain pair of categories are similar means that exists a strong correlation between them. Given a dataset $D = \{(\mathbf{f}_1, y_1), (\mathbf{f}_2, y_2), \ldots, (\mathbf{f}_n, y_n)\}$ over a domain of m labels $\mathcal{Y}$, there exist different approaches to infer an object-based category similarity measure $s(\lambda, \mu)$ between two labels $\lambda, \mu \in \mathcal{Y}$. According the work in this dissertation we distinguish two main groups of measures as followings.

- **Categorized similarity**. This approach considers the similarity measures between each pair of objects from different category to extract category similarity. Let $\mathbf{S} = (s_{ij})$ the $n \times n$ generic similarity matrix computed over D according any form of object-based similarity, we denote with $Q_{\lambda\mu}$ the set of real-valued similarity measures in $\mathbf{S}$ such that

$$Q_{\lambda\mu} = \{s_{ij} \mid y_i = \lambda \text{ and } y_j = \mu : i, j = 1 \dots n\}$$

The object similarities contained in the set $Q_{\lambda\mu}$ may be used to compute $s(\lambda, \mu)$ according several criteria as followings.

  - minimum: $s(\lambda, \mu) = \min(Q_{\lambda\mu})$
  - mean: $s(\lambda, \mu) = \dfrac{1}{|Q_{\lambda\mu}|} \sum_{s \in Q_{\lambda\mu}} s$
  - maximum: $s(\lambda, \mu) = \max(Q_{\lambda\mu})$

If $\mathbf{S}$ is asymmetric is not guarantee that the object-based category similarity $s(\lambda, \mu)$ is symmetric and its domain of definition depends by the original measure of object similarity employed.

- **Category centroid**. This approach is inspired to a well-known clustering task where a category of objects is represented by an unique reference feature vector said *centroid*; comparing such structures with respect to data from different categories it is possible to infer object-based category similarity. Formally the set of all feature vectors of a category $\lambda \in \mathcal{Y}$ may be defined as

$$Q_\lambda = \{\mathbf{f} \mid (\mathbf{f}, y) \in D, y = \lambda\}$$

The centroid of a category λ from D is obtained as the average feature vector

$$\mathbf{c}_\lambda = \frac{1}{|Q_\lambda|} \sum_{\mathbf{f} \in Q_\lambda} \mathbf{f}$$

For each pair of categories $\lambda, \mu \in \mathcal{Y}$ we can compute two forms of distance measures exploiting of the category centroids as followings.

  - centroid divergence: $d(\lambda, \mu) = \|\mathbf{c}_\lambda - \mathbf{c}_\mu\|$
  - visual distance[1]: $d(\lambda, \mu) = \dfrac{1}{|Q_\mu|} \sum_{\mathbf{f} \in Q_\mu} \|\mathbf{c}_\lambda - \mathbf{f}\|$

The distance measurement $d(\lambda, \mu)$ is always defined in $\mathbb{R}^+$ since it based on aggregated Euclidean distances and can be converted in similarity in different ways; the more general it is inspired to the classical rule (2.4) and may formalized for this context as followings

$$s^\gamma(\lambda, \mu) = \vartheta^\gamma\Big(d(\lambda, \mu)\Big)$$

[1]This form of distance is introduced in [14] considering objects of "visual" nature, nevertheless may be used in general terms for any extent of objects.

for a scaling positive parameter $\gamma \in \mathbb{R}^+$. The measures based on centroid divergence are symmetric, while this property is not guarantee for the visual distances.

2.2.4 Management of similarity measures

In this section we collect the main notions to define and manage measure of distance or similarity, in order to prepare such information for specific contexts of application.

2.2.4.1 Distance-based similarity

In machine learning there exist several measure of distance between data entities, but in many contexts it is necessary to convert and control them in an equivalent form of similarity, especially defined in the usual domain $[0, 1]$.

A wide employed tool that solves such aim is the famous Gaussian kernel, which is described as the following function:

$$\varrho^\sigma(d) = \exp\Big(- \frac{d^2}{2\sigma^2}\Big) \tag{2.3}$$

where $d \in \mathbb{R}^+$ is a distance measurement and $\sigma > 0$ is a positive parameter said Gaussian kernel width.

Let as usual the equation $d(\cdot)$ denotes some measurement of distance (e.g. euclidean, cosine, centroid based) the Gaussian similarity for two entities $\varepsilon_\alpha, \varepsilon_\beta \in \mathcal{E}$ is the function

$$s^\sigma_{gauss}(\varepsilon_\alpha, \varepsilon_\beta) = \varrho^\sigma\Big(d(\varepsilon_\alpha, \varepsilon_\beta)\Big)$$

This formulation present some properties:

- can be interpretable as a similarity measure just because decreases with the distance and ranges between 0 (since can not be negative in the limit) and 1 (when $\varepsilon_\alpha = \varepsilon_\beta$);
- when the distance measure is Euclidean the similarity measure matches with the well-known Gaussian radial basis function (RBF) kernel;
- the Gaussian function can be used as smoothness filter to convert a distance measure $d(\cdot)$ in similarity where the parameter σ is an useful tool for scaling and in aggregated similarities (see section 2.2.2.3).

Anyway the Gaussian kernel suggests a possible generalized form which allows equally to convert distance measures in similarities, but it is considered much

more flexible for scaling needs and may be employed in further contexts. This tool is described by the following rule:

$$\vartheta^{\gamma}(d) = \exp\Big(- \gamma d^2 \Big) \tag{2.4}$$

where $\gamma \in \mathbb{R}^+$ is a positive parameter to control the measure. Moreover it is interesting to observe that letting $\gamma = \dfrac{1}{2\sigma^2}$, the equation (2.4) reproduces the same behavior of the Gaussian kernel (2.3).

2.2.4.2 Symmetrization of affinity measures

The measure of similarity/dissimilarity between data entities is often assumed to be a symmetric operation, but in machine learning such property is not always respected. In fact in several methods to compute affinity measures, the produced mapping allows two different levels for the same pair of entities (e.g. Gaussian k-Nearest-Neighbor method, Resnik Similarity and so on). This is a unsuitable condition both for data analysis or even a wide class of computational algorithms, which requires to work over the assumption of symmetric relationships.

The problem consists to define rules to get symmetric an asymmetric measure. Considering to collect as usual all the possible measurements for a set of z data entities $\mathcal{E}$ in a typical similarity/distance $z \times z$ matrix $\mathbf{Z} = (z_{\alpha\beta})$ (which is asymmetric), the solution is acting an ideal operation of symmetrization generating a $z \times z$ structure $\hat{\mathbf{Z}} = (\hat{z}_{\alpha\beta})$, such that holds

$$\hat{\mathbf{Z}} = \hat{\mathbf{Z}}^T$$

Several possible ways to obtain this property could be the following assignment rules for each $\alpha, \beta = 1 \dots z$:

- minimum distance: $\hat{z}_{\alpha\beta} = \min(z_{\alpha\beta}, z_{\beta\alpha})$
- average affinity: $\hat{z}_{\alpha\beta} = \dfrac{z_{\alpha\beta} + z_{\beta\alpha}}{2}$
- maximum similarity: $\hat{z}_{\alpha\beta} = \max(z_{\alpha\beta}, z_{\beta\alpha})$

2.2.4.3 Normalization of similarity measures

Several measures of similarity are not defined in the typical domain $[0, 1]$ and this aspect in some context may be unsuitable. Considering to collect z data entities in a ideal set $\mathcal{E}$, whose the related similarity $z \times z$ matrix $\mathbf{S} = (s_{\alpha\beta})$ is obtained over a measure of similarity $s(\cdot)$ defined in $\mathbb{R}^+$, it is clear that the maximum value which may be assumed for each data entity α-th has to be $s_{\alpha\alpha}$. If this property holds, we may balance all the measures

to unity center applying the following general rule for each pair $\alpha, \beta = 1 \dots z$ as

$$\hat{s}_{\alpha\beta} = \frac{s_{\alpha\beta}}{s_{\alpha\alpha}} \tag{2.5}$$

to obtain a final collection of measures $\hat{\mathbf{S}}$ in $[0, 1]$, with $s_{\alpha\alpha} = 1$ for all $\alpha = 1 \dots z$. Clearly if $\mathbf{S}$ is asymmetric also $\hat{\mathbf{S}}$ continues to be that, but conversely it is not still guaranteed that $\hat{\mathbf{S}}$ remains symmetric; in fact may occurs that $\mathbf{S}$ has not an unique maximum similarity value, i.e. given some data entity $\varepsilon_\alpha \in \mathcal{E}$, $\exists \varepsilon_\beta \in \mathcal{E} \setminus \{\varepsilon_\alpha\} : s_{\alpha\alpha} \neq s_{\beta\beta}$, for this reason the rule (2.5) produces an asymmetric measure. When this condition represents a problem it is sufficient to apply some symmetrization rule (see section 2.2.4.2), which is in general based on maximum criterion.

2.3 Data Pre/Post Processing

The real computational algorithms employed in training tasks are not in general sufficiently flexible to work on any source of data, but are designed just according specif dataset structures and properties. Anyway an initial primitive collection of objects assembled for some aim could be registered in a document form which is less suitable for directed usages in machine learning. Moreover in this operational context are acted multiple steps of data preparation in such way to adjust both the form and the internal relations of the inputs which an algorithm expects.

In this section we introduce briefly some recurrent procedures of data pre/post processing indeed observed in this work.

2.3.1 Normalization/Standardization of descriptors

In section 2.1.3 we introduce the typical data model employed in machine learning under the assumption of object descriptors expressed as sequence of real-valued features. However if we consider a general point $\mathbf{f} = (f_1, f_2, \dots, f_q)^T \in \mathbb{R}^q$ may arise a crucial problem which is due to the domain of the measures related to each component with respect to the other. For example, if a feature f_k may have values limited in an interval of values particularly high (e.g. [100,200]) while all the other remain in another indeed small (e.g. [1,2]) it is clear that f_k is predominant. This fact can be a problem for the accurateness of affinity measures that depend by the length of the feature vectors (e.g. Euclidean distance in section 2.2.2). For example, if we consider a dataset $D = \{\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_n\}$ of n objects whose descriptors follow the schema as above, then in the computation of the distance between two objects $d_{euc}(\mathbf{f}_i, \mathbf{f}_j)$ just the information contained in the k-th component contributes effectively to the measure, while all the other features in a certain

sense are negligible.

The general method to solve this problem is well-known as Normalization/Standardization which allows to balance all the descriptors to share common properties from all the data D. There are two mainly formulations employed in the practice, where a feature vector $\mathbf{f}_i$ is replaced applying the followings rules.

- **Normalization**. All the descriptors are taken to the space $[0,1]^q$. Denoting with $\min(D) \in \mathbb{R}^q$ and $\max(D) \in \mathbb{R}^q$ respectively the minimum and maximum descriptors in D, a feature vector $\mathbf{f}_i$ is replaced by the rule:
$$\hat{\mathbf{f}}_i = \frac{\mathbf{f}_i - \min(D)}{\max(D) - \min(D)} \in [0,1]^q$$
- **Standardization**. All the descriptors have null mean and unitary standard deviation. Denoting with $\text{mean}(D) \in \mathbb{R}^q$ and $\text{dev}(D) \in \mathbb{R}^q$ respectively the average and standard deviation descriptors in D (in the section 2.4.4 are introduced the scalar versions for the same statistic concepts), a feature vector $\mathbf{f}_i$ is replaced by the rule:
$$\hat{\mathbf{f}}_i = \frac{\mathbf{f}_i - \text{mean}(D)}{\text{dev}(D)} \in \mathbb{R}^q$$

2.3.2 Image descriptors

In computer vision or image processing the objects of interest are clearly of visual nature. When the collection of data are set of images, the instances are very often furnished in the typical raster/bitmap format. In machine learning is more suitable to adopt the feature vector model (see section 2.1.3), therefore are requested techniques to convert an image in this other representation.

2.3.2.1 GIST descriptor

The well-know *GIST descriptor* [15] is a method to convert a raw image in a vector space of different dimensions, which is particularly addressed to scene recognition. The goal consists to produce a low depiction of a picture as an holistic representation of its spatial envelope, without required segmentation methods. The fundamental idea is based to define a set of perceptual dimensions (i.e. naturalness, openness, roughness, expansion and ruggedness) which represent the dominant spatial structure of a scene reliably estimable using spectral and coarsely localized information from Gabor filter response. In detail, the descriptor is produced as the energy output from a bank of Gabor filters, which are tuned on different orientations for z different scales as $\{o_1, o_2, \ldots, o_z\}$ (from high to low frequencies). Finally the square

output of each filter is averaged on a $n \times n$ grid. The resulting descriptor is a real valued feature vector $\mathbf{f}_{gist} \in \mathbb{R}^q$ whose number of dimensions is determined as

$$q = n^2 \sum_{s=1}^{z} o_s$$

2.3.2.2 Scale Invariant Feature Transform

The Scale Invariant Feature Transform (SIFT) is a particular method based on the research of special local points over an images which does not change (i.e. are invariant) under different transformations, such as rotation, scaling, illumination and different 3D camera viewpoints. Therefore these keypoints may be considered very distinctive features of an images and compared with other images allows to recognize common similarities. In other terms the SIFT descriptor depicts an image as a set of scale-invariant coordinates which are relative to local feature points. The process may be summarized in these following steps:

1. **Scale-space extrema detection**: under all the possible scales and image locations are searched the more interesting invariant points, the technique is based on difference of Gaussian function;

2. **Keypoint localization**: the keypoint are properly located according the measures of their stability;

3. **Orientation assignment**: for each keypoint is assigned r orientation histograms based on the gradient values is such point distributed in a grid of $n \times n$ subregions;

4. **Keypoint descriptor**: the information of the gradients around the keypoint are extracted and represented in a descriptor structure.

This particular process does not produce a conventional global feature vector of an image, but actually a set of different local descriptors, whose number k depends by the peculiarities of the input image processed. Anyway for each keypoint descriptor found during the transformation its dimension is determined as rn^2. Therefore the global SIFT descriptor of the i-th image may be seen as a matrix of size $rn^2 \times k_i$.

2.3.2.3 Spatial Pyramid Matching with Locality-constrained Linear Coding

The *Spatial Pyramid Matching* (SPM) [16] is a complex process to produce feature vector of images indeed adapted to scene and object classification. A wide class of descriptors (e.g. SIFT [17] or GIST) produce orderless bag-of-features depiction of images, which is not able to register properly the spatial

layout of the features, getting difficult to detect shape or segmenting objects in a background. The SPM method may be seen as a process which refines an initial object descriptor partitioning iteratively the image into increasingly finer spatial subregions where are computed statistics (or histograms) of local features. The typical setting consider sub-regions with a fixed form $2^l \times 2^l$ up to the limit $l = 0, 1, 2$. In detail the whole process may be divided in different main steps:

1. **Feature extraction**: keypoint locations/feature points on the image are detected or located in a fixed dense grid from which are extracted local descriptors (e.g. with SIFT method);

2. **Descriptor coding**: given a precomputed codebook/vocabulary of z entries (typically as a set of relevant centroids computed by k-means analysis [18] on a subset of images in the same domain of the picture of interest) each local descriptor is converted in a code of z dimensions which respects certain criteria (normally is used a VQ coding [19]).

3. **Spatial Pooling**: the feature points are analyzed for multiple levels of resolution and the sparse codes from each sub-region are pooled together by averaging and normalizing into a histogram.

4. **Concatenating**: the several histograms are concatenated together to produce the final global descriptor as a spatial pyramid representation of the image.

Denoting with l the max order of subdivision in the pooling with a codebook of z entries, the size of the produced descriptor $\mathbf{f} \in \mathbb{R}^q$ is

$$q = z \sum_{i=0}^{l} 4^i = z \frac{1}{3}(4^{l+1} - 1)$$

Although the VQ coding gives in general good performance, tends to produce descriptors quite sparse and hence less suitable for linear classifier (e.g. on typical Support Vector Machine (SVM) classifier). The Locality-constrained Linear Coding (LLC) [19] is a novel coding scheme for SPM application which exploits of locality constraint to project the descriptor into its local-coordinate system. Moreover the spatial pooling and concatenating phases do not use histograms since the results are combined with max pooling and normalization of the codes. The resulting LLC descriptor is be able to deal with linear classifier obtaining indeed better performances than VQ coding.

2.3.3 Normalized Graph Laplacian

This topic is especially addressed for algorithms which are designed to work on graph models. The well-know *Laplacian Matrix* is a particular

representation of a graph used to discover other new properties which are related to another graph structure, for example the calculation of its the spanning tree.

In machine learning this particular notion is widely used to improve an initial weighted adjacency matrix $\mathbf{W} = (w_{ij})$ generating its related Laplacian version $\hat{\mathbf{W}}$, which is said *normalized graph Laplacian*. Formally let $\mathbf{D} = (d_{ii})$ the diagonal degree matrix of $\mathbf{W}$ such that $d_{ii} = \sum_{j=1}^{n} w_{ij}$, the Laplacian matrix normalization is acted as followings:

$$\hat{\mathbf{W}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$$

Therefore the learning algorithm works actually on a input graph which has been updated with respect to initial definition. Anyway all the relationships contained in $\mathbf{W}$ are still maintained in $\hat{\mathbf{W}}$, but employing this normalization is considered as a good operation to enhance performance in real applications.

2.4 Data Analysis and Evaluation measures

In machine learning are indeed necessary performs data analysis on the training data or measuring the quality of an estimated model; for these tasks there exists several well-known solutions whose brief introduction is given in this section.

2.4.1 Similarity matrix Heat Map

A typical requirement in machine learning consists to evaluating the quality and accurateness of a similarity measure through a complete view of all the measurements, which may be related to objects, labels and any other entities of interest. Moreover in supervised applications it is fundamental to verify if the clusters organized according the similarity of the objects reflect the known labeling of the data, e.g. if it holds the concept of cluster assumption (see section 2.1.2).

Given an ideal set of z data entities $\mathcal{E}$ may be computed a $z \times z$ similarity matrix $\mathbf{S} = (s_{\alpha\beta})$ according some techniques (for example with centroid divergence in 2.2.3.2 or based on Gaussian k-Nearest-Neighbor method in 2.2.2.3). The main idea to analyze this structure consists to produce a visual depiction as a $z \times z$ bitmap map $\mathbf{P} = (p_{\alpha\beta})$ where each $p_{\alpha\beta}$ denotes an ideal color pixel. As concern the coloration aspect there exist different approaches. For example, denoting the set all possible colors as $\mathcal{C}$ is fixed a color map $c_{map} : [0, 1] \to \mathcal{C}$ (where the shade of colors are distributed according the minimum and maximum level of similarity) which is employed to compute for each pixel α and β the specific color as $p_{\alpha\beta} = c_{map}(s_{\alpha\beta})$. Another way

consists to generate a monochromatic depiction fixing a common color $c_0 \in \mathcal{C}$ and using an ideal operation $* : \mathcal{C} \times \mathbb{R} \to \mathcal{C}$ that compute the intensity of the color according the level of similarity between the entities, formally could be defined as $p_{\alpha\beta} = c_0 * s_{\alpha\beta}$. The heat maps can be useful to verify if the similarity measure is properly calibrated for the data or to detect regions interested by common similarity signals.

In the specif case of supervised learning the heat map could be employed to discover other important features among similarity measures and cluster membership. In this context the computation of the similarity matrix $\mathbf{S} = (s_{ij})$ occurs on a dataset of n objects $D = \{(\mathbf{f}_1, y_1), (\mathbf{f}_2, y_2), \ldots, (\mathbf{f}_n, y_n)\}$ over a domain of m labels $\mathcal{Y}$, where has to be respected a crucial arrangement of the data which concerns the disposition of the objects with respect to their supervised labels. The condition which is requested is that for each $i = 1 \ldots (n-1)$ if $y_i \neq y_{i+1}$ then $y_i \neq y_{i+k} : \forall k = (i+1) \ldots n$, in other terms the data are ordered for label.

You can observe that in this fashion along the principal diagonal of the matrix are distributed in sequence the m squared partitions with the pairwise similarities of points that share the same label, while the remained area denotes pairwise similarity between mismatched labels. The analysis of how appears the $n \times n$ image $\mathbf{P}$ can solve important answers:

1. if the partitions in the diagonal are indeed visible it means that objects of the same label are also quite similar among themselves;

2. if in the mismatching area there is less noise it means that objects of different labels are less similar;

3. if both 1 and 2 properties hold it means that according the similarity measure employed is verified the cluster assumption on the data;

4. if the picture is chaotic the reasons may be different, e.g. poor quality of the descriptors, a similarity measure unsuitable for the feature vectors, all the objects are too similar among themselves.

Exploiting of the supervised information of the data can be designed also another interesting way to define the nuance of the pixels. Usually for each label $\lambda \in Y$ is assigned a distinct color c_λ and is chose another color c_0 which is different from all the others too. The color for each pair of objects i and j is selected by the following rule

$$c(i,j) = \begin{cases} c_\lambda & y_i = \lambda = y_j \\ c_0 & \text{otherwise} \end{cases}$$

therefore the final color for each pixel can be express as

$$p_{ij} = c(i,j) * s_{ij}$$

This solution gets more evident along the main diagonal how is modeled the distribution of the membership block regions for each different label and the entity of similarity registered for all objects belonged to different clusters.

In practical usages the heat map of similarity matrix are a strong tool for data analysis, but the main weak of this method is just the visual approach, which is less suitable than a numerical measure of performance (e.g. the intra-class membership ratio in section 2.4.2). In fact when the number of labels are quite large the depiction of $\mathbf{P}$ can be difficult to consult.

2.4.2 Cluster membership

In the unsupervised learning, especially for the clustering approaches (see section 2.1.2), there exists different specific measures to evaluate the discovered clusters, in general the main kind of metrics are based on:

- **external index**: used to measure the degree of agreement between a known external labeling of the data and an estimated grouping discovered by a clustering analysis;
- **internal index**: used to measure the goodness of a clustering structure without using supervised external information.

In this dissertation we could formulate a coarse measure to evaluate how much a similarity measure $s(\cdot)$ between/among objects can reflect the labeling of the data points, in other terms the goodness of the cluster assumption. Given a dataset of n objects $D = \{(\mathbf{f}_1, y_1), (\mathbf{f}_2, y_2), \ldots, (\mathbf{f}_n, y_n)\}$ over a domain of m labels $\mathcal{Y}$ and let as usual the set of all objects in the same class $\lambda \in \mathcal{Y}$ as

$$Q_\lambda = \{\mathbf{f} \mid (\mathbf{f}, y) \in D, y = \lambda\}$$

we introduce the *intra-class membership ratio* for all the classes in $\mathcal{Y}$ as followings

$$m_{intra} = \frac{\sum_{\lambda \in \mathcal{Y}} \sum_{\mathbf{f}_i, \mathbf{f}_j \in Q_\lambda} s(\mathbf{f}_i, \mathbf{f}_j)}{\sum_{i,j} s(\mathbf{f}_i, \mathbf{f}_j)}$$

This measure may be exploited to formulate the global *inter-class membership ratio* clearly as

$$m_{inter} = 1 - m_{intra}$$

The cluster hypothesis says that objects similar should share the same label (or being in the same cluster), therefore the level of all the aggregated similarities for objects with the same labels with respect to all the possible pairwise similarities (i.e. intra-class membership ratio) should be quite high if holds this assumption (in fact, objects in different clusters should be not similar too); in other terms the hoped condition would be $m_{intra} \gg m_{inter}$.

2.4.3 Visualizing data in Scatter plot

In many data analysis the objects of interest are projected in a bi-dimensional or tri-dimensional scatter plot to obtain a visual overview of the data distribution in the Euclidean space. Therefore the objects are depicted as visual points whose distances reflect the similarity relation among them. This new way to see the data is indeed useful for different analysis both in unsupervised and supervised applications, for example: in the former may help to detect significant partition of the objects; in the latter the points may be colored per class producing a more detail depiction to verify cluster assumption.

The typical data model based on the representation of objects as multi-dimensional feature vectors (see section 2.1.3) may be disadvantageous when the feature space $\mathcal{F}$ exceeds the three dimensions. The solution of this problem consists to perform some method which can project the original data in a reduced space of two or at most three dimensions. Formally given an ideal dataset $D = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n\}$ of n objects on the feature space $\mathbb{R}^q$ with $q > 3$, we may model a process

$$\Phi^{(q \to r)} : \mathscr{P}(\mathbb{R}^q) \to \mathscr{P}(\mathbb{R}^r)$$

that produces a new dataset $P = \Phi^{(q \to r)}(D) = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ which contains the same n objects as reduced descriptors in the space $\mathbb{R}^r$ with $r = 2, 3$. The result obtained by this ideal technique has to be considered as a sort of global approximation P, which maintains most possible the same properties of the original data set D. Moreover if we consider an original instance $\mathbf{f}_i \in \mathbb{R}^q$, the reduced version $\mathbf{p}_i \in \mathbb{R}^r$ is not produced only according the local information of $\mathbf{f}_i$ but considering the peculiarities of all D too. Anyway it easy to observe that the operation of data reduction refers to a process much more general with respect to the specific application for visual representations, simply generalizing the case to the condition $r \ll q$. In this way the reduced data set P could be replaced in a learning phase to save computational time of execution.

In machine learning there exist several well-known techniques to reduce data dimensionality [20], but for visualization tasks are employed mainly Principal Component Analysis [21] and several variants of Multidimensional Scaling (MDS) [22] whose especially t-Distributed Stochastic Neighbor Embedding [23].

2.4.4 Standard deviation

In any context where are employed experimental measures obtained from aggregated data, there becomes necessary to establish an index to evaluate

their goodness. The mean of a set of values is the typical way to estimated a coarse measure which should represent the average case: knowing how much is the data variability can be an useful metric of precision for such expected value. Formally let $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}$ a generic set of n real-valued numbers where is computed the average measure as followings

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

the *standard deviation* (whose squared is said *variance*) is the typical dispersion index of the population X around $\bar{x}$, which is expressed as

$$\sigma_X = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n}}$$

If σ_X takes value quite high it means the estimation of $\bar{x}$ is made in a set of values with high variability, namely is rather imprecise. This indicator allows to say that the most part of the measures in X is roughly far of σ_X with respect to $\bar{x}$ or in other words that falls in the interval $[\bar{x} - \sigma_X, \bar{x} + \sigma_X]$. For example, if X denotes the age of a population with $\bar{x} = 25$ and $\sigma_X = 3$, then the most part of those persons are between 22 and 28 years old. Therefore is σ_X is low means that $\bar{x}$ represents better all the data in X, since there is a weak dispersion.

The standard deviation is often expressed as *coefficient of variation* in the normalized version

$$\sigma_X^* = \frac{\sigma_X}{\bar{x}}$$

which is more convenient since the value is always defined in $[0, 1]$ nevertheless X.

2.4.5 Classification error

In supervised learning the labeled dataset available D is used to sample two subsets $\left(D^{(train)}, D^{(test)}\right)$ said respectively *training* and *test* sets; according the type of inference (see section 2.1.2) with inductive learning the division is a partition, i.e. holds $D^{(train)} \cap D^{(test)} = \emptyset$, while in transductive learning the test cases are included in the training set but without the supervised information (as formalized in section 6.3.1). The data division is necessary for two different steps of whole process:

1. **Learning phase**. Estimation of a final mapper $\Psi_{D^{(train)}} : \mathcal{F} \to \mathcal{Y}$ between objects and labels according the input training set $D^{(train)}$ under (semi)supervised learning methods (see section 2.1.2).

2. **Evaluation phase**. Consists to determinate an error ratio committed by the model over the test set $D^{(test)}$.

The essential principle of the second step is based on mismatches between predictions and real response values provided by the testing individuals. Given a data split $D_s = \left(D^{(train)}, D^{(test)}\right)$ there exist two important measures of performances as followings:

- **Classification error rate**. It is a measure based on the rate of error produced by the model as:

$$e^{(D_s)} = \frac{|\{(\mathbf{f}, y) \in D^{(test)} \mid y \neq \Psi_{D^{(train)}}(\mathbf{f})\}|}{|D^{(test)}|}$$

- **Accuracy rate**: It is a measure based on the rate of success produces by the model as:

$$a^{(D_s)} = 1 - e^{(D_s)}$$

Normally the evaluation phase gives average measures of performances produced by much more divisions of the original dataset D, in this way the judgment of the model does not depends by a singular split chose. Formally defining as $\mathcal{S}_D$ the finite set of data splits, the mean classification error rate is expressed as

$$\bar{e}^{(\mathcal{S}_D)} = \frac{1}{|\mathcal{S}_D|} \sum_{D_s \in \mathcal{S}_D} e^{(D_s)}$$

and similarly as the general case, the mean classification accuracy is obtained as

$$\bar{a}^{(\mathcal{S}_D)} = 1 - \bar{e}^{(\mathcal{S}_D)}$$

The operational aspect to built $\mathcal{S}_D$ depends by the type of inference and what degree of precision is requested for the performances; in general there exist several well-known approaches to manage the validation, which are briefly described as followings:

- **Holdout**: producing a single split reserving $\frac{2}{3}$ for training and $\frac{1}{3}$ for testing with respect to D (the repartition is conventional only, other ratios are possible);
- **Repeated random sub-sampling**: producing several splits randomly and compute a mean measure of performances (if the repartition is fixed for all the splits the approach is said *Repeated Holdout*);
- k-**fold**: partitioning the data in k disjoint subsets, training on the $k-1$ partitions and testing in the remaining part, finally computing mean measure of all the performances (when $k = |D|$ the technique is said *Leave-one-out*);
- **Bootstrap**: producing data splits with replacement.

Chapter 3

Game Theory

The Game Theory is a science area modeled around a basic concept, which it can be easily explained quoting an essential statement by [1]:

> "The central problem of game theory was posed by von Neumann as early as 1926 in Göttingen. It is the following: if n players, $P_1, \ldots, P_n$, play a given game Γ, how must the i-th player, P_i, play to achieve the most favorable result for himself?"

The question gives us a clear intuition that the main aim of Game Theory consists to describe in formal way how a player can act to maximize own score, presumably on an environment subjected by the game rules and choices of own rivals. It is important to underline that the meaning of "game" has not to be considered in the typical playful terms, but it refers more generally to any problem which follows a finite schema.

The notions of Game Theory help to solve typical complex problems, such as Dominant Sets/Clustering/Segmentation [24], Graph Transduction [2], Standard Quadratic Programming problems (StQP) and Binary Maximal Cliques [25] and so on, which can be mapped to proper games whose solutions in this new form are easier to estimate than the original formulations. Thinking to be able to convert mathematical problems in games could be strange or difficult to understand, but if we consider a problem as a challenge between players the idea becomes more reasonable.

In this section we introduce the particular formalism and language of Game Theory field, focusing just about those main topics and aspects to understand our work.

3.1 Game assumptions

In Game Theory there exist different models of games, whose features depend mainly by the type of player interactions, the information that a player has with respect to the other players and what are his environment constrains to make an action and earn score. We introduce a famous subclass of games which is summarized by the following main properties.

- **Finite**. The number of players and the actions that they can exhibit are the first variables that determinate the "game dimension", which can be predetermined or not. Clearly when we talk about finite game, it means that is possible only a fixed number of players with a finite sets for their actions.

- **Non-cooperative**. The concept of cooperativeness in game theory matches with the idea of sets of players which interact among themselves giving life to coalitions. Such aspect is very strong to model some types of problems, wherein group of players can help each other to increase own payoffs. In the non-cooperative game instead we present an opposite scenario where the players are basically always independent (i.e. their actions does not constrained by those of the others) and enemies, with no possibility to tie relations.

- **Static/Simultaneous-move**. The typical game dynamics see rules which may often keep in account the sequential aspects of the player actions. The class of sequential games gives manner if a specific action is presented before or after another one. An important observation is the fact that the player has to wait the own turn before to make its decision, there exists trivially a cyclic order followed by players in the game play. Therefore the score gained by the players has to be modeled according all the possible action sequences. In this game is valid the assumption the player knows at least the last action made by the previous player which has played. Clarity this minimum level of knowledge for the players is necessary otherwise the time constrain would be not involve really to their payoff, since the game moves can not be described in sequential terms anymore. In fact, if there is lack of this information we are treating instead another famous game category. The simultaneous-move game depicts the democratic reality of a game whose the turn constrain is just absent, simply because all the players have to make own decision at the same time. As consequence of this property, we can say each player has not knowledge of the actions chosen by other players.

- **Complete and imperfect information**. A player can make decision mainly exploiting of the information he has about the playing field, in

particular with more relevance on response (just made and available) and score of own rivals: it is trivial, if we can see what move another player can do, it will involve of course on our future decision too. We are putting together two main concepts of Game Theory well-known as *completeness* and *perfection* of the information: the first one refers if a player knows the opposing available actions and payoff; the second one what particular replies the rivals have just made. The game we consider gives complete information, but as effect is also simultaneous-move the information is imperfect. In other terms is valid the assumption of *common knowledge*, it means all players known what all players known, and each player knows such information is shared to all.

- **Rational player**. The player is seen as an actor involved to make rational decisions. Anyway in this context the rationality property assumes only a specific meaning, in which the motivation that justifies the player move is the increasing on own payoff. There is no interest to model other possible goals that a player could follow during a game.

3.2 Normal-form game

The formal representation of a game it is not an easy task, since involves much and different fundamental information of the problem, such as players, actions, scores and relations among these elements; therefore it is necessary to adopt a general model to organized all this data. There exist two main forms to solve this aim, well-known as *Extensive-form* and *Normal-form games*. Although the former is able to generalize much more game schemes in this work we show in detail the latter only, since it is more suitable for the game assumptions as section 3.1.

We introduce the main objects needed to describe a game, the notations used is according [26].

- **Players**. Main components of a game is the set of its players, which can be formulated as $\mathcal{I} = \{1, 2, \ldots, n\}$ with $n \geq 2$. The definition of a single player $i \in \mathcal{I}$ in this form is quite general and it is necessary just to enumerate the several available actors. The classical meaning of "player" in this context does not be interpreted as a human individual, actually the effective nature of a player can be indeed different. Moreover it is not requested that all players in a game have to have the same playing freedom, but several roles are designable.

- **Pure Strategies**. The concept of pure strategy arises from the basic idea of action/move that an actor can exhibit. We can add new information for the player figure, which solves the need to assign those strategies

he can exploit. In formal way for each player $i \in \mathcal{I}$ is defined an own set $S_i = \{1, 2, \ldots, m_i\}$ of pure strategies.[1]

- **Pure Strategy profile**. The assumption of simultaneous-move gets simpler the formulation of a specific session of game; the term *profile* could be interpreted as synonymous of game stage and it can be described as a complete sequence with all the pure strategies whose players has played. This particular structure is called **pure strategy profile** modeled as a vector $\mathbf{s} = (s_1, s_2, \ldots, s_n)^T \in S$. This definition assumes it is clear and without ambiguities that the pure strategy at i-th position it is made by player $i \in \mathcal{I}$ and hence $s_i \in S_i$. The set $S = S_1 \times S_2 \times \ldots \times S_n$ (whose cardinality $|S| = \prod_{i=1}^{n} m_i$) is said **pure strategy profile space** and contains all the possible pure strategy profiles. We can observe that the game is finite if and only if the set S is finite.

- **Payoff function**. The fundamental tool needed to a game is an assignment of those scores the players have obtained in a certain game session. The payoff therefore depends both to specific player and pure strategy profile. The first aspect is trivial to explain, since each player has an own semantic role within the game, which involves the payoff that he deserves. The second one instead suggests something more important: to compute the payoff for a player i-th it is not necessary knowing only that pure strategy he has played, but also all those acted by your rivals. The domain of the score value could be defined in different forms, but very often is numerical and real. Made this assumption, for each player $i \in \mathcal{I}$ we define its personal payoff function as $\pi_i : S \to \mathbb{R}$. Given a pure strategy profile $\mathbf{s} \in S$, the payoff obtained by player i is the value $\pi_i(\mathbf{s})$. Moreover can be useful to define an extend payoff function which is able to compute the payoff of all the players as $\pi : S \to \mathbb{R}^n$. In this case we obtain directly all the payoffs as the vector $\pi(\mathbf{s}) = \Big(\pi_1(\mathbf{s}), \pi_2(\mathbf{s}), \ldots, \pi_n(\mathbf{s})\Big)^T$.

Therefore normal-form game is modeled as the triple $G = (\mathcal{I}, S, \pi)$, where $\mathcal{I}$ is the set of n players, S the pure strategy profile space and π the payoff function for all the players.

Presenting the game structure just by defining its sets of objects can be a way too strict for several applications. In Game Theory the normal-form games are famous for a special visual depiction of theirs features (with major

[1] We denote pure strategy as indexes for generality and suitable vectorial operations, but in other formulations it may be expressed as text labels; moreover, at least it is not by clear by the context, the pure strategies in different player sets are not semantically comparable, e.g. if given $h \in S_1$ and $k \in S_2$ holds $h = k$, it is not guaranteed that h and k refer to a same conceptual pure strategy.

interest to project the payoff function), which consists simply in a collection of tables or matrices. The easier case we introduce is that with two players, although it can be extended for greater games. In detail a two-player game can be described by two $m_1 \times m_2$ matrices, denotes as $\mathbf{A} = (a_{hk})$ for player 1 and $\mathbf{B} = (b_{hk})$ for 2, where for both the rows map the elements of the set of the strategies S_1 of the first player, while the columns the strategies S_2 of the second opposing player. If h and k are pure strategies such that $h \in S_1$ and $k \in S_2$, keeping the first matrix $\mathbf{A}$ as example, the a_{hk} cell contains the score assigned to player 1 in the pure strategy profile (h, k), in other terms $a_{hk} = \pi_1(h, k)$ $\Big(\text{hence } b_{hk} = \pi_2(h, k) \text{ for player } 2\Big)$.

The depiction of a game in matrix form can explain very quickly some special cases according certain properties between these structures, briefly we give the following definitions (assuming $m_1 = m_2$).

- **Zero-sum game**. When $\mathbf{A} + \mathbf{B} = \mathbf{0}$, hence it holds $a_{hk} = -b_{hk}$ for each $(h, k) \in S_1 \times S_2$.
- **Symmetric game**. When $\mathbf{A} = \mathbf{B}^T$, hence it holds $a_{hk} = b_{kh}$ for each $(h, k) \in S_1 \times S_2$.
- **Doubly-symmetric game**. When $\mathbf{A} = \mathbf{B}^T = \mathbf{A}^T$, hence the game is symmetric and further it holds $a_{hk} = a_{kh}$ for each $(h, k) \in S_1 \times S_2$.

Very often the formulation by couple of matrices appears very verbose and troublesome to use. In the practice it is sufficient an unique matrix expressible as $\mathbf{U}_{AB} = \mathbf{A} \cup \mathbf{B}$, which contains the information of the payoffs assigned to both player 1 and player 2. Each cell u_{hk} is the couple $(a_{hk}, b_{hk}) = \Big(\pi_1(h, k), \pi_2(h, k)\Big)$, we assume the order of what player payoff is the first or the second in the pair u_{hk} is clear by the context and respected for each cell of $\mathbf{U}_{AB}$.

3.2.1 Canonical games examples

Now we introduce briefly the schemes of some classic examples of games in normal-form, which are between two players and symmetric. Moreover in the section 3.4.3 we recall them analyzing in deep other aspects about the game solution.

Prisoner's dilemma
A typical simultaneous-move game is the *Prisoner's dilemma* which describes a controversial situation wherein one of two prisoners from the same criminal gang could betray the mate even if their mainly interest is to cooperate. The available and common action a prisoner can do is *confess* (betray the

mate admitting they are guilty) or *deny* the crime (cooperate with the mate remaining in silence). The consequences of that couple of actions occurred are the following:

- when a prison confesses but the other cooperates, the traitor is free while the other partner receives the full sentence of twenty months;
- if both deny the crime are sentenced for one-month sentence;
- if both confess are sentenced for ten-month sentence.

Using the game rules we can depict the related payoff matrix as followings:

		Prisoner 2	
		Confess	*Deny*
Prisoner 1	*Confess*	-10,-10	0,-20
	Deny	-20,0	-1,-1

Table 3.1: Payoff matrix of Prisoner's dilemma game

This version of the dilemma is generalizable, but we present it giving a possible practical scenario. Since game is symmetric, the preferable choice for a player is the same also considering the other party (i.e. *Confess*). The fundamental aspect of this problem consists that applying the assumption of rational player the solution of the problem is not that where both prisoner fall in the best situation, i.e. (*Deny*, *Deny*). The predicable human picture wherein two prisoners could cooperate to defend themselves is lost treating this situation as a normal-form game.

Rock-Paper-Scissors
Another typical simultaneous-move game is the *Rock-Paper-Scissors*, it requires just two players with the common set of pure strategies as {*Rock*, *Paper*, *Scissors*}. The symmetric rules of this game they refers to the features of the objects:

- an object again itself does not produce gain;
- the rock beats the scissors but loses with paper;
- the scissors beat the paper, but lose with rock;
- the paper beats the rock, but loses with scissors.

One possible depiction of the payoff matrix as the following:

		Player 2		
		Rock	*Scissors*	*Paper*
Player 1	*Rock*	0,0	1,-1	-1,1
	Scissors	-1,1	0,0	1,-1
	Paper	1,-1	-1,1	0,0

Table 3.2: Payoff matrix of Rock-Paper-Scissors game

Battle of the sexes
The *Battle of the sexes* game depicts the scenario wherein husband and wife have to meet them in some place, but they have forgotten the exact location. In detail both know the same set of possible places, moreover have some preferences. For example, if the locations are *football* and *opera*, it is assumed the husband prefers the first one, while the wife the second choice. Nevertheless the priority remains meeting in the same place.

Using the game rules we can depict the related payoff matrix as followings:

		Wife	
		Football	*Opera*
Husband	*Football*	2,1	0,0
	Opera	0,0	1,2

Table 3.3: Payoff matrix of Battle of the sexes game

The main interesting aspect of the problem is if both players make a wrong choice the score is equal and null, but conversely is unfair because one of the two players has always obtained a gain greater than the other.

3.3 Stochastic Normal-form game

The normal-form game schema introduced in the section 3.2 gives a basic and crisp formulation of a problem, but in other applications it is too rigid and incomplete to use. The main need consists to model that situation where the action of a player is ambiguous, i.e. multiple actions could be possible and with different weights. The formulation of the game therefore becomes extended in stochastic terms, so new objects have to be added to the pure model to treat this new concept.

- **Mixed strategy**. Given a player $i \in \mathcal{I}$ the idea of a move with multiple possibilities is described well by a probabilistic distribution over the set of pure strategy available S_i, this tool is called **mixed strategy** which is formulated as the vector

$$\mathbf{x}_i = \Big(x_i(1), x_i(2), \ldots, x_i(m_i)\Big)^T \in \Delta_{m_i}$$

  where the set Δ_{m_i} is the typical standard simplex which defines the domain of mixed strategies (it contains all the possible probabilistic distributions of m_i components, i.e. all mixed strategies of m_i pure strategies, for the player i). The single element $x_i(h) \in [0, 1]$ denotes the ratio that the pure strategy h happens for the player i. Moreover it is useful to define the **support** operator for a mixed strategy $\mathbf{x}_i \in \Delta_{m_i}$, which may be formalized as the function $\sigma(\mathbf{x}_i) = \{h \in S_i \mid x_i(h) > 0\}$, that returns the set of possible pure strategies (those could occur since their probability is not null). Anyway the definition of mixed strategy does not lost the base case, in fact a pure strategy is just a mixed strategy without uncertainty, in other terms the m_i vertexes of the standard simplex Δ_{m_i}. For example, the mixed strategy for the player $i \in \mathcal{I}$ which denotes exactly a pure strategy $h \in S_i$ is the point $\mathbf{e}_i^h \in \Delta_{m_i}$. It is fundamental to state that even if the mixed strategy object contains information that involves multiple moves, it has to be considered as an unique action employed by the player, but what it is in terms of pure strategy remains unsure.

- **Mixed strategy profile**. The concept of game *profile* remains clearly necessary also working with mixed strategies: we want to store the particular stage or point of a game. A **mixed strategy profile** therefore is a vector $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)^T \in \Theta$ which collects some possible mixed strategies by all the players, the component $\mathbf{x}_i \in \Delta_{m_i}$ is just the mixed strategy for the player i. Therefore the **mixed strategy profile space** is the set $\Theta = \Delta_{m_1} \times \Delta_{m_2} \times \ldots \times \Delta_{m_n}$.[1]

[1] For definition the standard simplex is an infinite set, hence Θ is infinite too; anyway the stochastic game has to be considered still finite, since it extends a finite pure strategy space S.

- **Payoff function**. The computation of the player score has to be updated to manage mixed strategy profiles. In this new model we have to build a payoff function $u_i : \Theta \to \mathbb{R}$ that gives the score of a player $i \in \mathcal{I}$ which has played in the mixed strategy profile $\mathbf{x} \in \Theta$. To model it we start from a strong assumption of probabilistic independence between the mixed strategies exhibited by all the players in a profile. Therefore given a pure strategy profile $\mathbf{s} = (s_1, s_2, \ldots, s_n)^T \in S$, the probability that may occur in a some mixed strategy profile $\mathbf{x}$ it is expressible as followings:

$$Pr(\mathbf{s}|\mathbf{x}) = \prod_{i=1}^{n} x_i(s_i)$$

The concrete payoff function is modeled as the expected value of the pure score weighted according the incidence of the pure strategy profile given into $\mathbf{x}$ of all the possible pure strategy profiles in S as:

$$u_i(\mathbf{x}) = \sum_{\mathbf{s} \in S} Pr(\mathbf{s}|\mathbf{x})\pi_i(\mathbf{s})$$

Even if such formula seems quite difficult to use, we can show its computation is quite generalizable. If we consider as usual the case of games with two players, given a mixed strategy profile $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)^T \in \Delta_{m_1} \times \Delta_{m_2}$ and recalling the two payoff matrix $\mathbf{A}$ and $\mathbf{B}$ of the pure model (see section 3.2), we can use simple products between these structures to solve calculus:

$$u_1(\mathbf{x}) = \sum_{h=1}^{m_1} \sum_{k=1}^{m_2} x_1(h) a_{hk} x_2(s_k) = \mathbf{x}_1^T \mathbf{A} \mathbf{x}_2$$
$$u_2(\mathbf{x}) = \sum_{h=1}^{m_1} \sum_{k=1}^{m_2} x_1(h) b_{hk} x_2(s_k) = \mathbf{x}_1^T \mathbf{B} \mathbf{x}_2$$

The stochastic model introduced can be depicted as an extension of the classic pure formulation $G = (\mathcal{I}, S, \pi)$ as the final schema $G^E = (G, \Theta, u)$, where Θ is the mixed strategy profile space and u is the ideal payoff function for all the players.

3.4 Toward the game solution

In Game Theory the definition of the problem in a certain form (e.g normal or extensive) is necessary mainly for the primarily task to describe in formal way what a game is. Nevertheless the next fundamental step consists to study what occurs in such environment: in other words, we are interested to look for the solution of the problem. In normal-form games it normally corresponds to specific game properties that strategies or strategy profiles can assume for some configurations and players.

Before to enter in deep about this topic we have to make some clarifications about the meaning of notation and language we are going to use. The strict distinctions between the terms *pure strategy* and *mixed strategy* as *pure strategy profile* and *mixed strategy profile* can be lightened using the general nomenclatures as *strategy* and *strategy profile*, because in practical refer always to the same basic concepts both in the pure and stochastic normal-form games; exclusively if the explanation can lead to awkward misunderstanding the distinction will be maintained. As concern the notation adopted we still follow the stochastic model introduced in the section 3.5, since it includes for its definition also the pure scheme properties (the reader can realize that the relations we are going to explain are applicable in one-to-one correspondence also with the objects of the pure normal-form game). Moreover we introduce other compacted objects to simplify some steps: given any strategy profile $\mathbf{y} \in \Theta$ and a mixed strategy $\mathbf{x}_i \in \Delta_{m_i}$, the object $\mathbf{z} = (\mathbf{x}_i, \mathbf{y}_{-i}) \in \Theta$ denotes the strategy profile wherein the player $i \in \mathcal{I}$ plays $\mathbf{x}_i$ while all the other players follow the strategies in $\mathbf{y}$. In other words the strategy $\mathbf{z}$ is modeled in such way $\mathbf{z}_i = \mathbf{x}_i$ for the $i \in \mathcal{I}$ and $\mathbf{z}_j = \mathbf{y}_j$ for each other player $j \in \mathcal{I} \setminus \{i\}$.

3.4.1 Best replies

We have to remember the assumption of rational decision-making player, i.e. his main aim is that to maximize own score. This crucial goal justifies the necessity to discover that strategy he should employ to reach it. The strategy $\mathbf{x}_i^* \in \Delta_{m_i}$ of the player $i \in \mathcal{I}$ is called **best reply/response** for the opponent strategy profile $\mathbf{y} \in \Theta$ if and only if it holds:

$$u_i(\mathbf{x}_i^*, \mathbf{y}_{-i}) \geq u_i(\mathbf{z}_i, \mathbf{y}_{-i})$$

for all the strategies $\mathbf{z}_i \in \Delta_{m_i} \setminus \mathbf{x}_i^*$. Therefore the best reply for a player i is just the choice that he has to do to obtain the greater score given a specific answer $\mathbf{y}$ by his rivals. In a game the set of all the possible best replies for the player $i \in \mathcal{I}$ against $\mathbf{y} \in \Theta$ is formulated as followings:

$$B_i(\mathbf{y}) = \left\{\mathbf{x}_i^* \in \Delta_{m_i} \mid \mathbf{y} \in \Theta, \forall \mathbf{z}_i \in \Delta_{m_i} : u_i(\mathbf{x}_i^*, \mathbf{y}_{-i}) \geq u_i(\mathbf{z}_i, \mathbf{y}_{-i})\right\}$$

In general the set $B_i(\mathbf{y})$ is always infinite, but it may occur a rare case wherein it exist a finite number of best replies, which are pure strategies too. Considering the sets of best replies for all the players in the common mixed strategy $\mathbf{y}$, it is possible to define a complete set of mixed strategy profiles as followings:

$$B(\mathbf{y}) = B_1(\mathbf{y}) \times B_2(\mathbf{y}) \times \ldots \times B_n(\mathbf{y}) \subset \Theta$$

Moreover we can underline two particular observation considering the relation between mixed strategies with the pure strategies in their support:

- if the support set of a best reply contains two or more pure strategies, any different mixed strategy with the same support is still a best reply;
- in the same way, if we detect two pure strategies which are best replies, any mixed strategies that support themselves are also best replies.

3.4.2 Nash equilibrium

The concept of best reply in section 3.4.1 focuses the incidence of a single strategy with respect to his author and the context where is applied. Very often we look for relevant properties that take in account all the set of players, especially for static games: the fundamental concept in Game Theory that looks the problem from this other point of view is well-known as **Nash Equilibrium**. This idea is a desired expectation under the assumption of rational decision-making for the players, which it should not be an inclination that lead to a game without a solution.

In formal terms the Nash Equilibrium is a particular strategy profile $\mathbf{x}^* = (\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_n^*) \in \Theta$ when is verified:

$$u_i(\mathbf{x}_i^*, \mathbf{x}_{-i}^*) \geq u_i(\mathbf{z}_i, \mathbf{x}_{-i}^*)$$

for all players $i \in \mathcal{I}$ and all the strategies $\mathbf{z}_i \in \Delta_{m_i} \setminus \mathbf{x}_i^*$. Moreover if the property is never equality for all $\mathbf{x}_i^* \neq \mathbf{z}_i$, the strategy profile $\mathbf{x}^* \in \Theta$ is defined as **strict Nash Equilibrium**, it means that $B(\mathbf{x}^*) = \{\mathbf{x}^*\}$.

It is easy to observe that Nash equilibrium exploits of the main concept of best reply extending it to all the players: in fact we could say it consists in a best reply for itself since $\mathbf{x}^* \in B(\mathbf{x}^*)$. Therefore when players reach a Nash equilibrium, they can do nothing to improve their gain changing moves. In a certain sense they all played in the best way considering their game profile, and such situation could be interpreted as sort of collective win or an equilibrium state.

We remark an important theorem which underlines the link of mixed Nash Equilibrium with the pure strategies:

Theorem 3.4.1 *Given a strategy profile $\mathbf{x}^* \in \Theta$ is a Nash Equilibrium if and only if for each player i, every pure strategy in the support of the mixed strategy $\mathbf{x}_i^*$ is a best reply with respect to the opponent strategies $\mathbf{x}_{-i}^*$. It follows that every pure strategy in the support of any mixed strategy in equilibrium leads its player to obtain the same expected payoff.*

3.4.2.1 Nash equilibrium existence

The condition about the existence of Nash equilibrium is a topic crucial in Game Theory. This particular strategy property has to be considered in

distinct way according if we consider the problem in terms of pure or mixed strategies.

- With *pure strategies* the only thing is possible to say is Nash Equilibrium could exist, therefore its existence is not guaranteed. Moreover a problem can admitting one or more Nash equilibrium, so such strategy profile is not necessarily unique.
- With *mixed strategies* instead the existence of at least one Nash Equilibrium is always guaranteed [3]. This property is fundamental because extending a pure game without Nash equilibrium in the mixed model, we can always find a potential solution for the same problem (although in stochastic terms).

3.4.3 Examples of Best Replies and Nash Equilibrium

The compacted payoff matrices of the games, at least for small dimensions, can be used to discover quickly best responses and Nash equilibrium. We can show some examples exploiting of our toy games in the section 3.2.1, where the **bold** style denotes best replies and the underlying for a profile in Nash equilibrium (anyway you can observe that for definition the latter matches to a cell which contains best replies for all the players).

Prisoner's dilemma

		Prisoner 2	
		Confess	*Deny*
Prisoner 1	*Confess*	**-10**,**-10**	**0**,-20
	Deny	-20, **0**	-1,-1

The unique Nash Equilibrium in this game, namely the profile (**Confess**, **Confess**), shows the interesting fact that is not the best situation for the prisoners, which it should be (*Deny*, *Deny*). The reason of this paradox is due to the assumptions of Game Theory, i.e. the rational decision-making of the player in a simultaneous-move situation, which may lead to a Nash equilibrium that fails the semantic of the game.

Rock-Paper-Scissors

		Player 2		
		Rock	*Scissors*	*Paper*
Player 1	*Rock*	0,0	**1**,-1	-1,**1**
	Scissors	-1,**1**	0,0	**1**,-1
	Paper	**1**,-1	-1,**1**	0,0

In this game does not exist a pure Nash equilibrium, but only a set of best replies. Anyway, according the properties of existence in section 3.4.2.1, using a mixed strategy extension we can solve the problem. In fact this game has an unique Nash equilibrium $(\mathbf{x}_1, \mathbf{x}_2) \in \Theta$ where the two players share the same uniform mixed strategy $\mathbf{x}_1 = \mathbf{x}_2 = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})^T$, i.e. the barycenter of Δ_3.

Battle of the sexes

		Wife	
		Football	*Opera*
Husband	*Football*	**2**,**1**	0,0
	Opera	0,0	**1**,**2**

The game admins even two Nash equilibrium, the profiles (*Football*, *Football*) and (*Opera*, *Opera*). The solutions reflect the condition of both players are addressed to make the same decision, anyway also in the equilibrium state a single player is always ahead.

3.4.4 Computing a Nash equilibrium

In real applications the research of Nash equilibrium is a challenging opened topic. It is necessary to understand in deep the nature of this formal problem and discover possible operational processes to solve it. The former aim has already reached a fundamental answer, classifying the Nash equilibrium in an important subclass of NP problems well-known as PPAD-complete [27]. Therefore we are dealing with a matter that can not be solved in a polynomial computational time. Fortunately we have available in literature different interesting approaches which suggests efficient algorithms to compute this solution, obtained by revised formulations of the main Nash Equilibrium concept. Considering the case of multiplayer games, we can cite for example the simplicial subdivision method [28], continuation methods [29] and enumeration-of-support methods [30, 31, 32].

In our work becomes indeed interesting to introduce a specific method which is derived from the well-known *Evolutionary Game Theory*, a sub disciple of the Game Theory whose fundamental contribute is ascribed to John Maynard Smith [26]. The game in this new area is modeled under special assumptions inspired on animal selection; the notions discovered through such curious formulation have opened the way to build concrete operational methods to compute Nash Equilibrium. This evolutionary approach is been furthermore enhanced by [33, 34] exploiting of the multipopulation idea to give a well-formed approach suitable on a multiplayer game.

The method to compute an equilibrium point is based on a particular

dynamical system (see section 2.1.3) called Replicator Dynamics. Modeling a mixed strategy profile as a function $\mathbf{x}^{(t)} \in \Theta$ in the domain of continuous time $t \in \mathbb{R}$ and considering a player $i \in \mathcal{I}$ which acts the pure strategy $h \in S_i$, the multiplayer version of the replicator is governed by the following relation:

$$\dot{x}_i(h) = x_i(h)\Big(u_i(\mathbf{e}_i^h, \mathbf{x}_{-i}) - u_i(\mathbf{x})\Big) \tag{3.1}$$

This particular dynamics belongs to a subclass more restricted included in the conventional regular selection dynamics based on *payoff monotonic dynamics*, which is featured by the property that the ratio of strategies with higher value of payoff tends to increase with higher rate [2]. The replicator is invariant when the payoff functions produce positive affine transformations during the evolution, i.e. $u_i(\mathbf{e}_i^h, \mathbf{x}_{-i}) \mapsto \alpha u_i(\mathbf{e}_i^h, \mathbf{x}_{-i}) + \beta$ with $\alpha > 0$. Moreover, if $\mathbf{x}^{(0)} \in \Theta$ is guaranteed for each player $i \in \mathcal{I}$ that $\mathbf{x}_i^{(t)} \in \Delta_{m_i}$ for all $t > 0$, namely each mixed strategy for the player i remains in its standard simplex Δ_{m_i}.

The fundamental property of the replicator dynamics is its fixed points are just Nash equilibrium, as stated in the following theorem [33]:

Theorem 3.4.2 *A point $\mathbf{x} \in \Theta$ is the limit of a trajectory of equation* (3.1) *starting from the interior of Θ if and only if $\mathbf{x}$ is a Nash equilibrium. Further, if point $\mathbf{x} \in \Theta$ is a strict Nash equilibrium, then it is asymptotically stable, additionally implying that the trajectories starting from all nearby states converge to $\mathbf{x}$.*

The formulation of the replicator dynamics as continuous function is a complete property from a theoretical point of view, but in real applications becomes indeed awkward. The typical algorithms have a vision of the time in a discrete domain, therefore we need to approximate the replicator dynamics to use it properly. The following operator is a good discretization in $t \in \mathbb{N}$

$$x_i^{(t+1)}(h) = x_i^{(t)}(h)\frac{u_i(\mathbf{e}_i^h, \mathbf{x}_{-i}^{(t)})}{u_i(\mathbf{x}^{(t)})} \tag{3.2}$$

which maintains the same dynamical properties of its continuous version (3.1).

Given the payoff function of all the players u and an initial mixed strategy profile $\mathbf{x}^{(0)} \in \Theta$ a possible algorithmic procedure to compute a Nash equilibrium is the following (where $\epsilon \in \mathbb{R}$ is a threshold of tolerance for the

updating process):

Algorithm 3.4.1: Discrete-Replicator-Dynamics($\mathbf{x}^{(0)}, u$)

$t \leftarrow 0$
repeat
▷ *computation of mixed strategy profile* $\mathbf{x}^{(t+1)} \in \Theta$
for $i \leftarrow 1$ **to** n
do $\begin{cases} s \leftarrow u_i(\mathbf{x}^{(t)}) \\ \textbf{for } h \leftarrow 1 \textbf{ to } m_i \\ \quad \textbf{do } x_i^{(t+1)}(h) \leftarrow x_i^{(t)}(h) \dfrac{u_i(\mathbf{e}_i^h, \mathbf{x}_{-i}^{(t)})}{s} \end{cases}$
$err \leftarrow \left\| \mathbf{x}^{(t+1)} - \mathbf{x}^{(t)} \right\|$
$t \leftarrow t + 1$
until $err < \epsilon$

return $(\mathbf{x}^{(t)})$

3.5 Succinct Games

In Game Theory literature there exists a wide collection of classical game patterns since different problems can be generalizable to unique schemes. Moreover an important feature taken in account for the categorization of these games is also their dimension, which is expressed in terms of number of payoffs needed to depict them. If we recall the classical normal-form model, without considering simplifications, a game with n players and m common pure strategies would require a set of nm^n values of payoff to be stored: we can realize it is not a trivial magnitude. To treat this problem we introduce the definition of **succinct game** (also known as **succinctly representable game**), which is a property of a game to be reducible in a lighter dimension than that it would be necessary with a normal-form representation.

We present very briefly a list of fundamental examples of succinct games.

- **Sparse games**. It is possible that the most part of the payoff values consists in a null number. This condition allows to exploit of a reduced depiction of the game considering the payoffs which are different by zero only; for example, with two player games the payoff matrices are sparse, therefore there could be used a typical *Compressed sparse row/column* schemes to store them in a reduced format.

- **Graphical games**. The impact on the payoff for a player could be involved to a restrict group of other players than all n. If we denote

with d the greater number of players which have indeed effect to the mates (presumably $d \ll n$), to depict the game are sufficient nm^{d+1} values of payoff. Moreover graphical games in a certain sense are a special case of sparse games.

- **Symmetric games**. The main aspect in these games consists that all players are identical, i.e. the rules of the game does not make distinction to that player is affected. For this reason the only thing really important to compute the payoffs in a possible profile is the number of n players that play each of the m strategies. The dimension of this type of game is just $m\binom{n+m-2}{m-1}$.

- **Anonymous games**. For each player is defined an own payoff function which does not depend to the extents of the other players (e.g. if the player has to choose to go in a given place, it is not relevant who will meet there, but how is crowded that location). For the calculus of the dimension of this game it holds the same concept of a symmetric game, but adding the payoff of the single player, hence the size is $nm\binom{n+m-2}{m-1}$.

- **Circuit games**. The idea consists to model the payoff function of a player as a logical combination of the opponent strategies (in other terms as a *Boolean circuit)*. Through this new form is not necessary to store each different output payoff because it can be computed on-the-fly.

We decide to dedicate a proper section for the explanation of another succinct game, since being that more relevant to understand our work it deserves a detailed treatment.

3.5.1 Polymatrix Games

The **polymatrix game** (also known as **multimatrix game**) is modeled according the assumption that the effect of any exhibited action from a player maintains the same incidence on the payoff for all the opponent players and independently by their moves. The important consequences of this property consists that only pairwise interactions between players are important. As concern the payoff computation for the player, since the influence of its strategy is the same for all the other ones, then it is solved simply as a sum of all the payoffs that its move produces against each opponent strategy.

We can give a formal definition of this game as following.

- Given a set of n players $\mathcal{I}$, all can play the same set of m pure strategy $\mathcal{P}$. There fore the pure strategy profile space is $S = \mathcal{P}^n$ and the mixed strategy profile space the multisimplex $\Theta = \Delta_m^n$.

- For each couple of players $i, j \in \mathcal{I}$ is defined a $m \times m$ payoff matrix as $\mathbf{A}_{ij} = \Big(a_{ij}(h,k)\Big)$. Moreover the interaction for each player $i \in \mathcal{I}$ with himself is useless to consider, hence we assume without lost generality that the payoff matrix $\mathbf{A}_{ii} = \mathbf{0}$, i.e. all its elements are set to zero.

- Given the pure strategy profile $\mathbf{s} = (s_1, s_2, \ldots, s_n)^T \in S$, the payoff function for a player $i \in \mathcal{I}$ is

$$\pi_i(\mathbf{s}) = \sum_{j=1}^{n} a_{ij}(s_i, s_j)$$

- Given the mixed strategy profile $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)^T \in \Theta$ the payoff function for a player $i \in \mathcal{I}$ is

  - for unambiguous mixed strategy

$$u_i(\mathbf{e}_i^h, \mathbf{x}_{-i}) = \sum_{j=1}^{n} (\mathbf{A}_{ij}\mathbf{x}_j)_h$$

  - for any mixed strategy

$$u_i(\mathbf{x}) = \sum_{j=1}^{n} \mathbf{x}_i^T \mathbf{A}_{ij}\mathbf{x}_j$$

In this class of games usually all the payoff partial matrices are collected in a unique block matrix structure $\mathbf{A} = (\mathbf{A}_{ij})$, whose magnitude reflects the dimension of the game too. In fact since we have a total of n^2 payoff matrices and each one contains always m^2 values, the size of a polymatrix game has order $O(n^2m^2)$.

The research of equilibrium points in this category of game clarity is an important case of study. Treating the formulation with stochastic structures is possible to guarantee the Nash equilibrium existence and in terms of computational complexity is been discovered that belongs to the class of PPAD-complete [35]. There exist two notorious algorithms to solve this game, which are based according different approaches to the same problem:

- the **replicator dynamic systems** derived by Evolutionary Game Theory (see 3.4.4);

- the **relaxation labeling operator** employed to find the solution of the *Labeling problem* (see 4.3).

Chapter 4

Labeling Problem

Several computational problems applied in different science fields such as, *computer vision*, *pattern recognition* and so on, fall in the main class of methods based on the necessity to give labels to a set of objects: this specific goal is generally well-known as *labeling problem*. It is presumable clearly that the association between label and object has to respect some semantic criteria. The latter refers to the fundamental *consistency* property of the assignments, which is obtainable combining several levels of information available on the objects. In literature there exist different processes to deal with this topics, but a wide visibility is certainly given to the well-known *consistent labeling problem* [36].

In this work we introduce a famous approach for labeling problem based on the *relaxation labeling process* formulated in 1976 by Rosenfeld, Hummel and Zucker [37]. The main aspect held up in this model is based on the combination of two levels of information associated to objects and predictable label. The *local* hypothesis for the labels that can be given to an object are weighted with the precious information of the *context*, which describes how much the forecasts are supported with respect to the assignments given to all the other objects. Looking for the best fitting that combines local and contextual information allows to reach a consistent labeling assignment. Moreover the relaxation labeling process may be seen as a generalization for another more general class of problems known as *Constraint Satisfaction problem* (CSP) [5, 6], wherein crisp constraints between variables are got "soft" by weights which denotes their level of confidence.

4.1 Discrete Binary CSP

The classical constraint satisfaction problem depicts a general set of crisp constraints on a set of variables which have to be totally satisfied. Therefore the valid solutions of the problem is just any assignments which respect all the constraints, for this reason they are also said *consistent assignments*. The *discrete binary* CSP is just a simplified case where the expressiveness of the conditions is more restricted. Formally a binary CSP can be modeled as the triple $(\mathcal{V}, \mathcal{D}, \mathcal{R})$ with the following features.

- $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of n variables.
- $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n\}$ is the set of *discrete* and *finite* domains associated to each variable, namely $\forall i = 1 \ldots n$, the variable $v_i \in \mathcal{D}_i$.
- $\mathcal{R} = \{R_{ij} \mid R_{ij} \subseteq D_{v_i} \times \mathcal{D}_{v_j}\}$ is the set of binary (or logical) constrains of each pair of variables, wherein $R_{ij} \in \mathcal{R}$ contains all legal pairs of values which may be taken by the variables v_i and v_j. Typically the depiction of a constrain R_{ij} can be formulates as a $|\mathcal{D}_{v_i}| \times |\mathcal{D}_{v_j}|$ binary matrix $\mathbf{R}_{ij} = \Big(r_{ij}(\lambda, \mu)\Big)$ where if the cell $r_{ij}(\lambda, \mu) = 1$ denotes that the assignment $v_i = \lambda \in \mathcal{D}_{v_i}$ is compatible with the assignment $v_j = \mu \in \mathcal{D}_{v_j}$. The case $r_{ij}(\lambda, \mu) = 0$ has the complementary meaning, i.e. such assignment for the two variables is incompatible (it is not permitted).

The formal definition of this problem gets clearer the reasons because is defined both *discrete* and *binary*: the domains of any variables are always finite and discrete; the constrains are logical expressions which evaluate the truthfulness or falseness on all the possible combinations of values which variables can taken. Hence this formulation is an instance of a general CSP problem, wherein are modeled instead any sort of variable domains and constrains.

Finding solutions in classical CSP over finite domains has a complexity time which falls in the class of NP-complete [38]. The trivial way to look for consistent assignment (or proof its nonexistence) is based on *backtracking*, but involves times too long for practicable usage; more suitable computational approaches are based on *local search* or *constraint propagation* [5, 6].

4.2 Relaxation Labeling model

The labeling problem and the process devised to solve it can be easily explained giving firstly a formal definition of all the conceptual elements and features which form the general model.

"**Problem**: let $\mathcal{B} = \{1, 2, \dots n\}$ a set of n objects and $\Lambda = \{1, 2, \dots m\}$ a set of m labels, the goal of the labeling problem is defining a mapping $\Psi : \mathcal{B} \to \Lambda$, where the label for each $i \in \mathcal{B}$ is solved as $\Psi(i) \in \Lambda$."

The initial information that has to be available for the objects are the following:

- **Local measurements**. The nature of the object involved in the problem have to allow its description in terms of measurable features (see section 2.1.3). This information is assumed to describe uniquely the object seen in isolation.
- **Contextual information**. According the possible nature of the problem it is possible to define a priori hypothesis which depicts all the knowledge of similarities between labels and objects. This information is collected in a ideal four-dimensional real-valued structure, which can be depicted as the $n \times n$ block matrix $\mathbf{R} = (\mathbf{R}_{ij})$ (whose extended magnitude is $nm \times nm$), wherein for each couple of objects $i, j \in \mathcal{B}$ is stored the partial $m \times m$ matrix $\mathbf{R}_{ij} = \Big(r_{ij}(\lambda, \mu)\Big)$. Given a pair of labels $\lambda, \mu \in \Lambda$, the real value $r_{ij}(\lambda, \mu)$ is called **compatibility coefficient** and it measures the strength of the following hypothesis:

  "the object i is labeled with λ" and "the object j is labeled with μ".

  In general high values for a compatibility coefficient denotes the hypothesis is very likely, and low values vice versa; the condition $r_{ij}(\lambda, \mu) \geq 0$ is preferable in several applications, but not necessary. How inferring the matrix $\mathbf{R}$ properly is an open problem.

  The formulation just introduced gets evident that relaxation labeling may be seen as a soft version of the Discrete Binary CSP, wherein the variables match with the objects which share the same domain of labels and especially the crisp logical constrains are taken in a stochastic model.

4.3 Relaxation Labeling process

The method to solve the labeling problem is obtained by a relaxation process wherein, starting from an initial assignment, it will be iteratively updated up to reaching the final solution. For each step $t \in \mathbb{N}$ and object $i \in \mathcal{B}$ it is necessary to compute a structure named **local weighted labeling assignment** which collects the hypothesis of all the possible labels; formally it can be expressed as the following vector:

$$\mathbf{p}_i^{(t)} = \Big(p_i^{(t)}(1), p_i^{(t)}(2), \dots, p_i^{(t)}(m)\Big)^T \in \Delta_m$$

In other words $\mathbf{p}_i^{(t)}$ is a probabilistic distribution over the set of labels Λ in the standard simplex Δ_m. For each label $\lambda \in \Lambda$ the value $p_i^{(t)}(\lambda)$ measures the weight of the hypothesis:

"the object i is labeled with λ at time t".

Therefore if $\mathbf{p}_i^{(t)}$ is a vertex point of the standard simplex Δ_m, it means which depicts an unambiguous assignment because it exists a possible associated label only. The local measurements of an object i may be exploited just to solve the problem to infer a proper initial assignment $\mathbf{p}_i^{(0)}$ at time $t = 0$. There exists interesting techniques to deal with this problem, for example in image processing the information of a the color histogram describes the distribution of colors in an image, if is found a relevant percentage of blue shades for a picture the hypothesis its label may be *sky* is reasonable high. Anyway in several applications, if this information is too poor to give a proper initial assignment is acted a general initialization as the barycenter of the standard simplex, i.e. $\mathbf{p}_i^{(0)} = \left(\frac{1}{m}, \cdots, \frac{1}{m}\right)^T \in \Delta_m$.

Considering the whole set of n objects we can define another combined structure called **weighted labeling assignment**, defined in the multi-simplex space $\mathbb{K} = \Delta_m^n$ as followings:

$$\mathbf{p}^{(t)} = \left(\mathbf{p}_1^{(t)}, \mathbf{p}_2^{(t)}, \ldots, \mathbf{p}_n^{(t)}\right)^T \in \mathbb{K}$$

Similarly the case of single object, when $\mathbf{p}^{(t)}$ is a vertex point of the space $\mathbb{K}$, it depicts an unambiguous weighted labeling assignment since for each object there is no uncertainty of its associated label.

Another dynamical structure needed in the updating process for each object $i \in \mathcal{B}$ at time t is called **support vector** and modeled as followings

$$\mathbf{q}_i^{(t)} = \left(q_i^{(t)}(1), q_i^{(t)}(2), \ldots, q_i^{(t)}(m)\right)^T \in \mathbb{R}^m$$

where the element for each label $\lambda \in \Lambda$ is computed as

$$q_i^{(t)}(\lambda) = \sum_{j=1}^{n} \sum_{\mu=1}^{m} r_{ij}(\lambda, \mu) p_j^{(t)}(\mu) \tag{4.1}$$

The contextual information given by $q_i^{(t)}(\lambda)$ could be interpreted as a measure for the statement "the context supports that object i is labeled with λ at time t". According the definition for a single element (4.1), exploiting of matrix operation the complete support vector for an object i can be expressed equally in the following form:

$$\boldsymbol{q}_i^{(t)} = \sum_{j=1}^{n} \mathbf{R}_{ij} \mathbf{p}_j^{(t)} \tag{4.2}$$

As the weighted labeling assignment, the global support vector of all objects at time t is denoted as $\mathbf{q}^{(t)} \in (\mathbb{R}^m)^n$.[1] Further for the proposed definition (4.2), it holds that $\mathbf{q}^{(t)} = \mathbf{R}\mathbf{p}^{(t)}$.

We introduce a heuristic solution well-known in honor of its authors as Rosenfeld-Hummel-Zucker rule, which reflects the combination of local and contextual information over the objects.

For each object $i \in \mathcal{B}$ and label $\lambda \in \Lambda$, the assignment $\mathbf{p}_i^{(t)}$ is updated at time $t+1$ as followings:

$$p_i^{(t+1)}(\lambda) = \frac{p_i^{(t)}(\lambda) q_i^{(t)}(\lambda)}{s} = \frac{p_i^{(t)}(\lambda) q_i^{(t)}(\lambda)}{\sum_{\mu=1}^{m} p_i^{(t)}(\mu) q_i^{(t)}(\mu)} \tag{4.3}$$

The rule simply exploits of the two main information weights about the label λ and the object i, whose nature is local for $p_i^{(t)}(\lambda)$ and contextual for $q_i^{(t)}(\lambda)$, normalized according the sum s for each labels to guarantee that $\mathbf{p}_i^{(t)} \in \Delta_m$.

Relaxation labeling in operational terms is an iterative process which from an initial weighted labeling assignment $\mathbf{p}^{(0)}$ generates a sequence of updated versions $\mathbf{p}^{(0)}, \mathbf{p}^{(1)}, \mathbf{p}^{(2)}, \dots \mathbf{p}^{(t)}$ along the time t, with the goal to converge in a fixed point for the rule (4.3). This final weighted labeling assignment depicts the solution (soft) of the labeling problem. A possible algorithmic form which describes this process may be summarized as followings (where $\epsilon \in \mathbb{R}$ is a threshold of tolerance for the weighted labeling assignment updating).

[1] This shortcut notation has be not confused as $(\mathbb{R}^m)^n = \mathbb{R}^{mn}$, but the Cartesian power is meant as $(\mathbb{R}^m)^n = \underbrace{\mathbb{R}^m \times \dots \times \mathbb{R}^m}_{n \text{ times}}$.

Algorithm 4.3.1: Relaxation-Labeling-Process($\mathbf{p}^{(0)}, \mathbf{R}$)

$t \leftarrow 0$
repeat
  $\triangleright$ *computation of support vector* $\mathbf{q}^{(t)} \in (\mathbb{R}^m)^n$
    and weighted labeling assignment $\mathbf{p}^{(t+1)} \in \mathbb{IK}$
  for $i \leftarrow 1$ **to** n

$$\textbf{do} \begin{cases} s \leftarrow 0 \\ \textbf{for } \lambda \leftarrow 1 \textbf{ to } m \\ \quad \textbf{do} \begin{cases} q_i^{(t)}(\lambda) = \sum_{j=1}^{n} \sum_{\mu=1}^{m} r_{ij}(\lambda, \mu) p_j^{(t)}(\mu) \\ p_i^{(t+1)}(\lambda) = p_i^{(t)}(\lambda) q_i^{(t)}(\lambda) \\ s \leftarrow s + p_i^{(t+1)}(\lambda) \end{cases} \\ \mathbf{p}_i^{(t+1)} \leftarrow \dfrac{1}{s} \mathbf{p}_i^{(t+1)} \end{cases}$$

  $err \leftarrow \left\| \mathbf{p}^{(t+1)} - \mathbf{p}^{(t)} \right\|$
  $t \leftarrow t + 1$
until $err < \epsilon$

return $(\mathbf{p}^{(t)})$

Clearly if the process finishes at the final labeling $\mathbf{p} \in \mathbb{IK}$, it is very difficult it is an unambiguous assignment too. A classical idea to get crisp the soft assignments is based to select the most likely label for each object $i \in \mathcal{B}$. Hence the estimated crisp labeling assignment can be modeled as a vector

$$\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n)^T \in \Lambda^n$$

where for each object $i \in \mathcal{B}$ the outcome is obtained by the typical maximum *a posteriori* probability (MAP) rule

$$\hat{y}_i = \arg \max_{\lambda \in \Lambda} p_i(\lambda)$$

Recalling the initial formulation of the problem, the estimated mapping is therefore:

$$\Psi(i) = \hat{y}_i \quad \forall i = 1, \ldots, n$$

4.4 Consistency and Dynamics of Relaxation Labeling

The relaxation labeling process may be formalized as an ideal continuous mapping $\mathcal{L} : \mathbb{IK} \to \mathbb{IK}$, where $\mathbf{p}^{(t+1)} = \mathcal{L}(\mathbf{p}^{(t)})$ with $t \geq 0$ according the rule (4.3), or in other terms as a discrete dynamical system (see section 2.1.3).

An open question consists to determinate what conditions have to hold to guarantee the reaching of an equilibrium state for the dynamical operator $\mathcal{L}$, namely if exists $t \in \mathbb{N}$ finite such that $\mathcal{L}(\mathbf{p}^{(t)}) = \mathbf{p}^{(t)}$. This topic is been discussed in different studies [4, 39, 40] and we report the fundamental properties involved in this process.

Since the updating of weighted labeling assignment depends uniquely by its support vector, the relaxation evolves just if its contextual truthfulness changes. If $\mathbf{p} \in \mathbb{K}$ is a equilibrium point for $\mathcal{L}$ and $\mathbf{q} \in (\mathbb{R}^m)^n$ is its support vector, then has to hold the following relation:

$$q_i(\lambda) = c_i \text{ whenever } p_i(\lambda) > 0 \quad \forall i = 1, \ldots, n, \ \lambda = 1, \ldots, m$$

for a some positive constants $c_1, c_2, \ldots, c_n$ (we can realize from this property that an unambiguous weighted labeling assignment is hence an equilibrium point for $\mathcal{L}$, even if it is not always true the opposite).

The particular property as above can be formalized in another form which is well-known as Hummel and Zucker's consistency [4] of the weighted labeling assignment. Let as usual $\mathbf{p} \in \mathbb{K}$ and $\mathbf{q} \in (\mathbb{R}^m)^n$ the related support vector, then $\mathbf{p}$ is said **consistent** if:

$$\sum_{\mu=1}^{m} p_i(\mu) q_i(\mu) \geq \sum_{\mu=1}^{m} z_i(\mu) q_i(\mu) \quad \forall i = 1, \ldots, n$$

for all the possible weighted labeling assignment $\mathbf{z} \in \mathbb{K}$. Moreover if relation is never equality for each $\mathbf{z} \neq \mathbf{p}$, the weighted labeling assignment $\mathbf{p}$ is said **strictly consistent** (e.g. the unambiguous points). The consistency condition is equivalent to the vectorial property $(\mathbf{z} - \mathbf{p})^T \mathbf{q} \leq 0$ for all $\mathbf{z} \in \mathbb{K}$, from which we derive a geometrical interpretation: when the support vector becomes orthogonal with respect to all the tangent vectors of $\mathbf{p}$, it is reached the consistency.

The main theorem and a relative corollary which summarize all this concepts about to consistency property are the following [4].

Theorem 4.4.1 *A weighted labeling assignment $\mathbf{p} \in \mathbb{K}$ is said consistent if and only if, for all $i = 1, \ldots, n$ and $\lambda = 1, \ldots, m$, hold the two conditions:*

1. $q_i(\lambda) = c_i$ *whenever* $p_i(\lambda) > 0$
2. $q_i(\lambda) \leq c_i$ *whenever* $p_i(\lambda) = 0$

for some positive constants $c_1, c_2, \ldots, c_n$.

Corollary 4.4.2 *Let $\mathbf{p} \in \mathbb{K}$ be consistent, then $\mathbf{p}$ is a fixed point for the nonlinear relaxation operator $\mathcal{L}$. If $\mathbf{p}$ is in the interior of $\mathbb{K}$ space the converse also holds.*

The support vector depends mainly by the fixed weights of the compatibility coefficients in the matrix $\mathbf{R}$. Therefore it is presumable that the properties assumed on such structure involve heavy to the relaxation labeling process. We said that the matrix $\mathbf{R}$ is symmetric when for all $i, j \in \mathcal{B}$ holds the condition $\mathbf{R}_{ij} = \mathbf{R}_{ji}^T$. To explain what occurs on the basis of $\mathbf{R}$ we have to introduce the particular notion of **average local consistency** of an weighted labeling assignment $\mathbf{p} \in \mathbb{IK}$, which is depicted by the function $\Upsilon : \mathbb{IK} \to \mathbb{R}$ as following:

$$\Upsilon(\mathbf{p}) = \sum_{i=1}^{n} \sum_{\lambda=1}^{m} p_i(\lambda) q_i(\lambda) = \sum_{i=1}^{n} \mathbf{p}_i^T \mathbf{q}_i = \mathbf{p}^T \mathbf{q} \tag{4.4}$$

We quote another fundamental theorem from the Hummel and Zucker work [4] which describes the behavior of the average local consistency under certain conditions.

Theorem 4.4.3 *Suppose that the compatibility matrix $\boldsymbol{R}$ is symmetric. Then any local maximum $\boldsymbol{p} \in \mathbb{IK}$ of Υ is a consistent weighted labeling assignment.*

This result is really important because under such assumption on $\mathbf{R}$ the consistent labeling task can be turned into a maximum problem of a function, or better the following StQP:

$$\begin{array}{ll} \text{maximize} & \Upsilon(\mathbf{p}) \\ \text{subject to} & \mathbf{p} \in \mathbb{IK} \end{array}$$

Moreover if $\mathbf{R}$ is symmetric and positive the Rosenfeld-Hummel-Zucker rule works increasing strictly the amount of average consistency [40], in other terms it assumes the typical role of Lyapunov function for the system (see section 2.1.3). The evolution of the whole process is stable and finite since it holds the following relation

$$\Upsilon\Big(\mathcal{L}(\mathbf{p}^{(t)})\Big) > \Upsilon\Big(\mathbf{p}^{(t)}\Big)$$

for each $t \geq 0$, until it will be reached a fixed point for the operator $\mathcal{L}$.

In general we can not presume that with an asymmetric $\mathbf{R}$ the process never converges, in fact with unambiguous points it is always finite. Anyway we can state that in presence of a strictly consistent labeling $\mathbf{p} \in \mathbb{IK}$ the condition of symmetry for the compatibility matrix is irrelevant, since $\mathbf{p}$ is also an asymptotically stable equilibrium point for the nonlinear relaxation operator $\mathcal{L}$ [39, 40].

4.5 Relaxation labeling in Game Theory

The labeling problem model in the section 4.3 is conceived according assumptions which give space to a large set of applications. In our work it

is important to introduce the contribute of relaxation labeling in the Game Theory field. The interesting fact is the similarity between the problem to look for the solution of a **Polymatrix game** (see section 3.5.1): in fact we can say that is possible to define an one-to-one correspondence between them [41].

The mapping of these two problems can be easily understandable by a comparison as following.

Relaxation Labeling	***Polymatrix Game***
Set of n objects $\mathcal{B}$	Set of n players $\mathcal{I}$
Set of m labels Λ	A unique set of m pure strategies $\mathcal{P}$ shared for all the players
Compatibility coefficients $\mathbf{R}_{ij}$ for a couple of objects $i, j \in \mathcal{B}$	Payoff matrix $\mathbf{A}_{ij}$ for a couple of players $i, j \in \mathcal{I}$
Block matrix $\mathbf{R} = (\mathbf{R}_{ij})$ which collects all partial compatibility coefficients	Block payoff matrix $\mathbf{A} = (\mathbf{A}_{ij})$ which collects all partial payoff matrices
Weighted labeling assignment $\mathbf{p}_i \in \Delta_m$ for the object $i \in \mathcal{B}$	Mixed strategy $\mathbf{x}_i \in \Delta_m$ for the player $i \in \mathcal{I}$
Weighted labeling assignment $\mathbf{p} \in \mathbb{K}$	Mixed strategy profile $\mathbf{x} \in \Theta$
Consistent labeling assignment $\mathbf{p} \in \mathbb{K}$	Nash Equilibrium $\mathbf{x} \in \Theta$
Strictly consistent labeling assignment $\mathbf{p} \in \mathbb{K}$	Strictly Nash equilibrium $\mathbf{x} \in \Theta$

Table 4.1: Comparison of Relaxation Labeling and Polymatrix Game

The fundamental connection between these apparently different problems is just the property of a consistent labeling assignment matches with the Nash equilibrium of a mixed strategy profile. Therefore relaxation labeling may be used as an useful computational method to find Nash equilibrium points in any polymatrix game. In fact it is observable that the nonlinear relaxation labeling rule (4.3) may be seen as the explicit form that the general replicator dynamics (3.2) would assume according the payoff function of this game class (see section 3.4.4).

Chapter 5

Graph Transduction Problem

In the machine learning community semi-supervised approaches to deal with labeling tasks (see section 2.1.2) are already well-consolidated methods and the interest about these topics is in continuous ascent nowadays [9]. A wide class of semi-supervised applications defines a theoretical problem as a graph model, wherein the objects of interest coincide to nodes and the weights over the edges are similarity measures among themselves (see section 2.2.2): one of these graph-based techniques is known as *Graph Transduction* method. In that the main matter to face consists to look for a suitable way to propagate the supervised information given by known labeled nodes to unlabeled ones, in order to obtain a consistent labeling for all the objects, which is an approach in harmony with the typical transductive inference approach (see section 2.1.2). It is important to state that this specific class of semi-supervised learning works under the fundamental assumption on the data well-known as *clustering assumption*, which reflects the natural *homophily* principle of the social networks[1]: namely, the tendency of individuals to associate and bind with *similar* others [42].

In this work we introduce a novel solution of Graph Transduction, which models the problem as a multiplayer noncooperative game [2] exploiting of a strong relationship between the relaxation labeling (see chapter 4) process and notions in Game Theory field (see section 4.5).

5.1 Graph Transduction model

First of all it is necessary to give a formal definition of the problem to have a clear look of this topic. Graph transduction is a method which is based on semi-supervised learning process. Therefore we can image to have

[1]Refers to an important branch of the social sciences, particularly in sociology and anthropology, which studies the theoretical social structure made up of individuals or organizations.

a set $\mathcal{B} = \{1, 2, \ldots, n\}$ of n objects which can be labeled in the finite domain $\mathcal{Y} = \{1, 2, \ldots, m\}$ of m labels. For each object $i \in \mathcal{B}$ is associated a proper feature vector $\mathbf{f}_i$ (see section 2.1.3) to describe its isolated peculiarities in an ideal global domain $\mathcal{F}$ for all the objects. The concrete training set available can be modeled in the form $D = (D_\ell, D_{-\ell})$ where:

- $D_\ell = \{(\mathbf{f}_1, y_1), (\mathbf{f}_2, y_2), \ldots, (\mathbf{f}_k, y_z)\}$ is the sub set of $z \ll n$ labeled objects where $\forall i = 1 \ldots z$ the label $y_i \in \mathcal{Y}$;
- $D_{-\ell} = \{\mathbf{f}_{z+1}, \mathbf{f}_{z+2}, \ldots, \mathbf{f}_n\}$ is the sub set of $n - k$ unlabeled objects;

Moreover we introduce these shortcut notations $\mathcal{B}(D_\ell) = \{1, 2 \ldots z\}$ and $\mathcal{B}(D_{-\ell}) = \{z + 1, z + 2, \ldots, n\}$ to denote respectively the sub sets of labeled and unlabeled objects within of all the population $\mathcal{B}$.

The information in the dataset D which is available for all the individuals it is clearly their feature vectors. These descriptors can be employed to infer a graph structure according some methods (see section 2.2.2) to obtain a $n \times n$ weighted adjacency matrix $\mathbf{W} = (w_{ij})$ which contained pairwise similarity scores of the objects. Moreover it is necessary that $\mathbf{W}$ has to be assumed as a null-diagonal matrix, since it is useless modeling self-loops for the Graph Transduction problem. Therefore exploiting of the similarities in $\mathbf{W}$ the construction of the weighted graph $\mathcal{G} = (\mathcal{B}, E, \mathbf{W})$ is devised according these following aspects:

- $\mathcal{B}$: is the set of n labeled and unlabeled objects which may be seen without lost generality as nodes/vertexes of the graph too;
- $E = \{(i, j) \in \mathcal{B}^2 : w_{ij} \neq 0\}$: is the set of edges in the graph which depicts the relevant relationships among objects;
- $\mathbf{W}$: the similarity matrix whose $\mathcal{G}$ is inferred represents just its weighted adjacency matrix, which further may be seen as an ideal function $w : E \to \mathbb{R}$ that returns the real-valued weight of an edge $(i, j) \in E$ as

$$w(i, j) = w_{ij}$$

The goal of the Graph Transduction consists to estimated a final general mapper $\Psi : \mathcal{B} \to \mathcal{Y}$, under cluster assumption in the training set D, which furnishes a consisting labeling for all the nodes/objects.

5.2 Devising a transductive learning

The research of a possible solution for the Graph Transduction problem may begin under the analysis of a simplified case of study, which is obtained imposing certain conditions on the input graph structure. We can image to

have an instance of the dataset D such that produces a graph $\mathcal{G}$ which is an *unweighted undirected graph*; in this way the related structure $\mathbf{W}$ is just a binary adjacency matrix (i.e. defined in the domain $\{0, 1\}$) and symmetric (i.e. $\mathbf{W} = \mathbf{W}^T$) since the graph is undirected too. Hence the presence of an edge between two nodes denotes the *perfect* similarity between the related pair of objects, while conversely their total dissimilarity: basically, two objects can be either completely equal or different, without intermediate weights. Under this setting the fundamental clustering assumption gains a new interpretation, since in this simplified graph a data cluster in D matches with a connected component (i.e. a subgraph of $\mathcal{G}$ in which any pair of nodes is connected), therefore it is equal to say that:

"nodes in the same connected component should share the same label".

An interesting way to solve this toy problem could be giving its reformulation as a discrete binary constraint satisfaction problem (see section 4.3). In fact if we consider to depict the label assigned to each node as a variable, following the cluster assumption we derive that the information in the adjacency matrix $\mathbf{W}$ is sufficient to define all the constrains that have to be satisfied by the variables. Formally all the elements associated to this CSP can be defined as followings:

- **Variables.** Since the graph is composed by n nodes the set of variables is simply determined as $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$.
- **Variable domains.** The domains for the labeled objects $\mathcal{B}(D_\ell)$ are restricted to their unique outcome value, while all the others may assume any of the m labels available. Hence the generic domain for a variable $v_i \in \mathcal{V}$ is defined as

$$\mathcal{D}_{v_i} = \begin{cases} \{y_i\} & i \in \mathcal{B}(D_\ell) \\ \mathcal{Y} & i \in \mathcal{B}(D_{-\ell}) \end{cases}$$

- **Constrains.** It is interesting to observe that given a pair of nodes $i, j \in \mathcal{B}$, if $w_{ij} = 1$ means that the two objects are equal, therefore for the cluster assumption should have the same label too: this condition is translated to the constrain $v_i = v_j$. Recalling the formulation of the constrain set as the $|\mathcal{D}_{v_i}| \times |\mathcal{D}_{v_j}|$ binary matrix $\mathbf{R}_{ij} = \Big(r_{ij}(\lambda, \mu)\Big)$, to model $v_i = v_j$ it is sufficient that for each pair $(\lambda, \mu) \in \mathcal{D}_{v_i} \times \mathcal{D}_{v_j}$ the element $r_{ij}(\lambda, \mu) = 1$ just when holds $\lambda = \mu$. You can observe that for each unlabeled variables/objects $i, j \in \mathcal{B}(D_{-\ell})$ this definition produces exactly $\mathbf{R}_{ij} = \mathbf{I}_m$, namely the compatibility matrix is equal to the identity matrix of $m \times m$ order. Trivially if $w_{ij} = 0$ (the two objects are different) the pair of variables v_i and v_j are free to assume any values, i.e. their assignments has not be constrained between themselves, which

is reflected by the condition $r_{ij}(\lambda, \mu) = 1$ for any value λ and μ in their domains. All these indications can be summarized by the following rule:

$$r_{ij}(\lambda, \mu) = \begin{cases} \begin{cases} 1 & \lambda = \mu \\ 0 & \text{otherwise} \end{cases} & w_{ij} = 1 \\ 1 & \text{otherwise} \end{cases} \tag{5.1}$$

Therefore any assignments which satisfy all the constrains modeled on this discrete binary CSP are consistent labeling assignments, or equally valid solutions for the toy graph transduction problem considered.

5.3 Graph Transduction Game

The weak point of the transductive approach introduced in section 5.2 is due to the crisp constrains modeled by CSP formulation, which are not able to deal with any general graph, i.e. when $\mathbf{W}$ is defined in $\mathbb{R}$. Hence it would be necessary lookin for a way to maintain the constrain satisfaction principle but that is *softer* too. The relaxation labeling model (see section 4.3) may arise as a sharp source of inspiration since describes just that is requested. Moreover another aspect much more interesting is that in Game theory field (see chapter 3) there exist several notions that allow to model the same transductive process: hence it is reasonably possible to formulate the Graph Transduction problem as a noncooperative game.

The n objects in the dataset D, or in other terms the nodes of the associated graph $\mathcal{G}$, may be translated as players of a normal form game (see section 3.2), expressed by the set $\mathcal{I} = \{1, 2, \ldots, n\} \equiv \mathcal{B}$. The labels in $\mathcal{Y}$ can be treated as pure strategies for the individuals. Therefore all the players share the same set of m pure strategies $\mathcal{P} = \{1, 2, \ldots, m\} \equiv \mathcal{Y}$ (namely, for each player $i \in \mathcal{I}$ is assigned $S_i = \mathcal{P}$), hence the pure strategy profile space is the set $S = \mathcal{P}^n$. The interpretation of the game consists that a player $i \in \mathcal{I}$ which plays a pure strategy $h \in \mathcal{P}$ declares the own membership for that associated label. Anyway this pure formulation is still too rigid to model our goal, therefore the current game is extended in a stochastic model (see section 3.5). All the players can play mixed strategies, which clearly are defined in the common mixed strategy space as the standard simplex of m components Δ_m. The derived mixed strategy profile space is therefore the multisimplex $\Theta = \Delta_m^n$. Moreover the complete set of players associated to labeled objects can be formulated as $\mathcal{I}_\ell \equiv \mathcal{B}(D_\ell)$ where the remained parts is denotes as $\mathcal{I}_{-\ell} \equiv \mathcal{B}(D_{-\ell})$. According the knowledge of the problem is clear that each player does not act at the same way. In fact it is reasonable that a player associated to a labeled object $i \in \mathcal{I}_\ell$ can play always the unique pure strategy k associated to its supervised label $y_i \equiv k$, or equally the related unambiguous mixed strategy $\mathbf{e}_i^k$. Therefore we denote with $\mathcal{I}_\ell \mid k \subset \mathcal{I}_\ell$

the subset of these labeled players that play the common pure strategy $k \in \mathcal{P}$.

Treating the problem to define payoff functions we can assume that only pairwise interactions between players are allowed. Therefore a suitable reference schema which may be adopted it is that of a typical Polymatrix game (see section3.5.1). Anyway the general definition of this class of games can be explicated in this context highlighting the specific distinction between the contribute given by labeled and unlabeled players. Formally, for each player $i \in \mathcal{I}$ on a given mixed strategy profile $\mathbf{x} \in \mathbf{\Theta}$, the payoff functions in this polymatrix game are reformulated as followings:

- For unambiguous mixed strategy

$$u_i(\mathbf{e}_i^h, \mathbf{x}_{-i}) = \sum_{j \in \mathcal{I}_{-\ell}} (\mathbf{A}_{ij}\mathbf{x}_j)_h + \sum_{k=1}^{m} \sum_{j \in \mathcal{I}_\ell | k}^{n} \mathbf{A}_{ij}(h,k)$$

- For any mixed strategy

$$u_i(\mathbf{x}) = \sum_{j \in \mathcal{I}_{-\ell}} \mathbf{x}_i^T \mathbf{A}_{ij}\mathbf{x}_j + \sum_{k=1}^{m} \sum_{j \in \mathcal{I}_\ell | k}^{n} \mathbf{x}_i^T (\mathbf{A}_{ij})_k$$

The final question to deal with is how to define the partial payoff matrix $\mathbf{A}_{ij}$ for each pair $i, j \in \mathcal{I}$ of players. To emulate the same transductive principle drafted in the section 5.2 it is necessary to look for a general model, which can turn the logical crisp constrains in a soft form in such way the cluster assumption setting is maintained. In that is interesting to realize the weight of the hypothesis which two players have the same label may be expressed just as much they are similar, in other terms through their similarity measure. Therefore the general formulation of the partial payoff matrix for the players $i, j \in \mathcal{I}$ becomes:

$$\mathbf{A}_{ij} = w_{ij}\mathbf{I}_m$$

The assumption of null-diagonal condition for the weighted adjacency matrix $\mathbf{W}$ produces null payoff matrices $\mathbf{A}_{ii}$ since holds $w_{ii} = 0$ and this consequence is in harmony with respect to the standard formulation of a polymatrix game. Moreover the typical organization of all the payoff partial matrices can be taken in the unique block matrix $\mathbf{A} = (\mathbf{A}_{ij})$ which is easily computed by Kronecker product as followings:

$$\mathbf{A} = \mathbf{W} \otimes \mathbf{I}_m \tag{5.2}$$

We can observe that such model in the case of unweighted undirected graph reproduces just the constrains formulation (5.1) for the case $w_{ij} = 1$, which is more evident for unlabeled players/objects, i.e. $\mathbf{A}_{ij} = w_{ij}\mathbf{I}_m = 1\mathbf{I}_m = \mathbf{R}_{ij}$.

In fact if the game is reduced to pure strategy playing, it emulates just the same discrete binary CSP behavior.

This noncooperative game is singular since the players $\mathcal{I}_\ell$ basically plays always the same strategies, therefore their aim is not to maximize own payoff: in a certain sense they act as prompters for the concrete active players $\mathcal{I}_{-\ell}$, in such way the latter are led to perform those strategies which reflects better their membership. Although it is clear that this principle have to be maximized globally, which matches to that state wherein all the players reach the best fitting simultaneously; hence the solution of this game is just represented by the well-known concept of Nash equilibrium (see section 3.4.2.1).

5.4 Operational settings of GTG

The transductive process of the Graph Transduction Game (GTG) has been analyzed mainly as a formal model in section 5.3. Considering the solution as a computational problem instead there arises new questions about a proper data management and the consequent dynamical behavior of the learning phase.

It is necessary to decide how prepare the initial state of the dynamical system associated to GTG. Since this game is formulated for semi-supervised learning, they are known without ambiguities the labels associated to the objects in the subset D_ℓ, while for the rest of the data $D_{-\ell}$ this lack of information should be solved by the Nash equilibrium computed. As concern the initial mixed strategies to give to all player, the favorite way consists to assign for a labeled object the unambiguous strategy according its supervised label, while the barycentric of the common standard simplex Δ_m over $\mathcal{Y}$ for the unlabeled objects. Formally given a player $i \in \mathcal{B}$ the rule applied to define the mixed strategy $\mathbf{x}^{(0)} \in \Delta_m$ is the following:

$$\mathbf{x}_i^{(0)} = \begin{cases} \mathbf{e}^{y_i} & i \in \mathcal{I}_\ell \\ \left(\frac{1}{m} \cdots \frac{1}{m}\right)^T & i \in \mathcal{I}_{-\ell} \end{cases}$$

The payoff block matrix $\mathbf{A}$ is initialized according the rule (5.2) and such information is a parameter crucial of whole the learning process, since depicts a fixed context along all the learning; whereas it depends mainly from the weighted adjacency matrix $\mathbf{W}$ of the graph $\mathcal{G}$, therefore the quality of such information is indeed important. In that there are several aspects to consider: the accurateness of the descriptors in $\mathcal{F}$; the similarity technique to compute $\mathbf{W}$; the fundamental cluster assumption on the data. It is interesting to observe another fact: if $\mathbf{W}$ is symmetric also the payoff matrix $\mathbf{A}$ becomes symmetric. It has been amply argued about how the symmetric

property of the compatibility coefficients determines specif behaviors on the dynamical process which is acted to compute equilibrium points (see theorem 4.4.3). Nevertheless to the original procedures to obtain $\mathbf{W}$, it is followed a typical practice known in several machine learning applications of Laplacian normalization (see section 2.3.3). In detail, before to begin the training phase is precomputed the normalized graph Laplacian $\hat{\mathbf{W}}$ according the given similarity matrix $\mathbf{W}$, therefore the real payoff block matrix modeled by GTG algorithm is actually:

$$\hat{\mathbf{A}} = \hat{\mathbf{W}} \otimes \mathbf{I}_m.$$

Assuming that in an instance of Graph Transduction Game is found a concrete Nash equilibrium $\mathbf{x} \in \Theta$, there remains the problem about how to express this consistent weighted labeling assignment as crisp labels predicted for all the players. The classical solution (as suggested for the relaxation labeling model in section 4.3) is based to infer that pure strategy stronger for each player $i \in \mathcal{I}$ in terms of maximum *a posteriori* probability by the following rule

$$\hat{y}_i = \arg\max_{k \in \mathcal{P}} x_i(k) \tag{5.3}$$

which selects the more likely label to assign for the associated object $i \in \mathcal{B}$. This final step depicts the consistent mapper Ψ as the solution of Graph transduction problem (see section 5.1).

The Graph Transduction Game is mainly a type of polymatrix game which can be solved by relaxation labeling process (see section 4.3) or in evolutionary terms by the replicator dynamics (see section 3.5.1). Considering the real implementation of this game the concrete transductive learning is formalized always as a discrete dynamical system. In this setting, since a polymatrix game is mappable perfectly with a relaxation labeling process, GTG inherits the same dynamical behaviors of Hummel and Zucker's system (see section 4.4). The computational time of the learning process, shared by both the discrete dynamics (3.2) and (4.3), is estimable in the order $\mathcal{O}(tmn^2)$, where as usual n is the number of players/objects, m is the number of pure strategies/labels and t the number of steps required to converge in a Nash equilibrium point/Consistent labeling assignment. Another important consideration consists to realize that if the payoff matrix $\mathbf{A}$ is symmetric, then it is also guaranteed that the transductive process has a finite time; anyway even if such condition does not hold the algorithm could however reach a fixed point, which for the theorems 3.4.2 or 4.4.1 is still a Nash equilibrium. Nevertheless in general does not exist a formal law to describe the number of iterations t required for the learning, anyway it has been observed a linear increasing with respect to the number of objects n. This latter consideration takes Graph Transduction Game to follow the computational time of $\mathcal{O}(n^3)$, which is on the line with the trends of other popular graph-based transduction

methods. For this reason GTG would be not suitable for on-line learning applications (see section 2.1.2); anyway there exist possible studies which may be exploitable to increase the current performance, for example a novel fast evolutionary game dynamics [43] or the MUCCA algorithm [44], both solutions with linear time complexity.

5.5 Introducing Category similarity in GTG

Typical classifiers consider each single category completely different in terms of similarity with respect to the others. However this assumption is not always suitable in different applications, because may occur that some class is for a certain aspect similar to another one, e.g. *street* and *highway*, *pony* and *horse*, *car* and *jeep* and so on. Especially when the number of categories is quite high, the class similarity sharing clearly is not negligible anymore and hence there becomes indeed indispensable to introduce category similarity in the learning process. The approach to exploit of class similarities may be modeled in several manners, but in general it is based to weight properly the similarity among objects from different categories. In other terms the category similarity should help to improve accurateness whose a computer system distinguishes multi-categorized objects [45].

In this dissertation we are interested to study the behavior of GTG when its underlying transductive principle based on cluster assumption is extended to incorporate category similarity information during the learning. The challenging fact being able to formulate a classifier as above is clearly hoped, but we treat this topic mainly for analysis tasks. For such purpose, we may devise a presumable model which generalizes the original formulation of the Graph Transduction Game. We remark that GTG is built under a relaxation scheme wherein the possible labeling assignments are modeled as a set of soft constrains. Nevertheless the general pattern that defines the high level similarity relationships among labels (as for example the linguist semantic or object-based forms of category similarity, see section 2.2.3) is a binary structure which does not consider the real domain of categories involved, since it rejects *a priori* possible levels of similarity between couple of different labels.

In formal way we recall firstly some fundamental data components employed in GTG. Given a data set D of n objects $\mathcal{B}$, it is inferred a graph structure $\mathcal{G} = (\mathcal{B}, E, \mathbf{W})$ whose $n \times n$ null-diagonal weighted adjacency matrix $\mathbf{W} = (w_{ij})$ contains for each couple $i, j \in \mathcal{B}$ of nodes/objects their similarity measure w_{ij}. The domain of labels that the objects may assume is a finite set of m categories $\mathcal{Y}$. For each edge $(i, j) \in E$ in $\mathcal{G}$ is associated a $m \times m$

matrix that models all the soft constrains for the labeling assignments as

$$\mathbf{A}_{ij} = w_{ij}\mathbf{I}_m$$

which matches with the general payoff function for the polymatrix Graph Transduction Game. You can observe that the binary matrix factor $\mathbf{I}_m$ may be seen just as the similarity matrix which models the category similarities over the domain $\mathcal{Y}$, but actually it is independent by the real categories. In fact it remains constant for any set $\mathcal{Y}$, scaling on m only; moreover since $\mathbf{I}_m$ is always an identity matrix, it depicts that particular case where a set of m categories does not share similarity, i.e. any label $\lambda \in \mathcal{Y}$ may be completely equal only to itself $s(\lambda, \lambda) = 1$, but totally different with respect to all the other $s(\lambda, \mu) = 0 : \forall \mu \in \mathcal{Y} \setminus \{\lambda\}$ (considering typical similarity measures in $[0, 1]$). This aspect is presumable to be the main weak point of GTG and such limit suggests the definition of a further generalized form as

$$\mathbf{A}_{ij} = w_{ij}\mathbf{S}^{(\mathcal{Y})}$$

where the structure $\mathbf{S}^{(\mathcal{Y})} = (s_{\lambda\mu})$ represents the real-valued $m \times m$ similarity matrix over the category domain $\mathcal{Y}$ and $s_{\lambda\mu}$ weights the affinity between two labels $\lambda, \mu \in \mathcal{Y}$ (which may be computed by different class similarity measures as section 2.2.3). Moreover we could assume as suitable precondition that $\mathbf{S}^{(\mathcal{Y})}$ is symmetric and with all ones along the main diagonal, as holds for a wide class of similarity measures; it is reasonable that such properties should weak much less possible the underlying cluster hypothesis and avoid ambiguous relationships between categories. The collection of payoffs for any pairs of objects is collected in the new block matrix as

$$\mathbf{A} = \mathbf{W} \otimes \mathbf{S}^{(\mathcal{Y})}$$

We call this new scheme *Graph Transduction Game with Category Similarity* (GTGwCS) which combines object-based similarity $\mathbf{W}$ with category similarity $\mathbf{S}^{(\mathcal{Y})}$ just with a Kronecker product between these two kinds of information. This new formulation does not lost generality since when $\mathbf{S}^{(\mathcal{Y})} = \mathbf{I}_m$, then GTGwCS reproduces the same behavior of the original GTG.

Chapter 6

Experimental survey of GTG

Our analysis is mainly well-established on fundamental assumptions related to image recognition in Computer Vision. Considering a general categorization problem, we could say that the successful of such task depends if the objects grouped in a given category reflect its semantic, therefore there becomes interesting to wonder an essential thing: "What is a category?". In cognitive psychology [46] categories are seen as semantic units in the human mind, which are so important to be considerable as the basis of human intelligence [47]. In this cognitive science the open questions focus about how humans can define categories and how they are depicted in the mind. Moreover another fundamental research consists to verify if there exist conceptual/semantic prototypes associated to a category [48]. If we move our point of view considering the interaction of the human in the visual world, the cognitive psychology supports different important relations between semantic category and visual similarity of images, in detail:

- semantic categories are visually separable, or in other terms form cluster in visual space;
- there exist possible visual prototypes which describe a semantic category;
- visual similarity is correlated to semantic similarity.

Therefore taking this topics in computer vision, we are interested to verify if a machine can respect those assumptions. In other terms, we wonder if the common similarity measures used to distinguish images are able to reflect the semantic differences of the categories involved too: in fact the greater parts of the algorithms which employ visual affinities consider implicit such ability. We solve our doubt from an interesting analysis [14], which proofs that typical similarity measured by computer vision descriptors is sufficiently able, under some preconditions, to follow the semantic bound with the categories in the cognitive human visual system. This fact is fundamental otherwise we could

not use computer vision tools to design solutions for image recognition tasks.

In this section we take these concepts in a real experience made through the Graph Transduction Game (GTG) (see section 5), to understand in detail what reasons are behind the measured results. We start introducing general topics about practical aspects which are common for the greater part of all the experiments, in particular:

- the main peculiarities concerning the datasets employed and their technical depiction in feature vectors;
- the effective method to infer a graph structure from a dataset and the several parameters/properties established;
- the evaluation strategies used to compute and compare performances.

Then we eventually analyze in deep several aspects about Graph Transduction Game, from preliminary observations to other sharper surveys focusing the relationships between visual and semantic similarities. Finally we complete our study employing the variant GTGwCS (see section 5.5) to see what occurs in GTG if measures of category similarities are introduced in the learning process.

6.1 Datasets

Graph Transduction Game is designed over similarity relationships without to impose what sort of objects is involved. In the original work [2] GTG was already tested in different kind of datasets, which collect images, text documents, citation networks, web pages and so on. Anyway in our practical experiences we employ several dataset especially of visual nature whose main features are introduced as followings.

- **Scene** [15]. Scene is set of 2688 natural scene images with resolution of 256×256 pixels, which are classified into eight different categories. Moreover for each class there are an average of 335 scenes.
- **SUN** [49]. SUN is a comprehensive collection of annotated images covering a large variety of environmental scenes, places and the objects within (131,072 Images, 908 Scene categories, 249,522 Segmented objects and 3819 Object categories). Our dataset consists in a selection of 397 scene categories, where the number of images for each category is variable with a minimum support of 100 examples for a total of 108,754 images.
- **Caltech-101** [50]. Caltech-101 is a collection of 9144 images which are distributed in 101 categories. The subjects depicted in each class

are objects (e.g. cars, keys, flowers, animals and so on) with relevant variance in terms of shape. For each category are available about 40 to 800 images, but the major part there contains 50 items. The resolution of each image is roughly of 300×200 pixels.

6.1.1 Object descriptors

For all the datasets employed the images are given as JPEG documents, the conversion phase to vectorial object descriptors is described as followings.

- For the datasets Scene and SUN the original images are resized to be not larger than 256×256 pixels (preserving the aspect ratio) and then converted with GIST descriptors (see section 2.3.2.1). In detail it is followed the original configuration [15] employing bank of 24 Gabor filters for 8 orientations and 4 scales, whose signals are arranged in a grid of 4×4 blocks: the resulting GIST descriptor is a feature vector of 512 dimensions.

- For the dataset Caltech-101 the original images are resized to be not larger than 300×300 pixels (preserving the aspect ratio) and then converted through Spatial Pyramid Matching method with Locality-constrained Linear Coding (see section 2.3.2.3). The initial feature points in an image are extracted with SIFT approach (see section 2.3.2.2) with 8 orientation histograms on 4×4 subregions producing local descriptors in 128 dimensions. A dictionary of 1024 entries is generated by k-means clustering [18] over a subset of objects with an uniform number of images for class. The spatial max pooling is performed over 4×4, 2×2, and 1×1 sub-regions. Therefore the final LLC descriptor for each images has 21,504 dimensions.

According the peculiarities of the descriptors employed, the feature vectors produced have not predominant features. On the basis of experimental evaluations the process of Normalization/Standardization of the datasets (see section 2.3.2) is been avoided, since relevant differences in term of performance has not been observed.

6.2 Graph construction

The Graph Transduction Game works in a graph structure inferred from an initial input dataset, hence it is crucial to define the better setting to maintain faithful information with respect to the original input points. In all the experiments we perform a graph construction technique that is inspired from the original work [2] which we reintroduce the main aspects.

The usual weighted adjacency matrix $\mathbf{W} = (w_{ij})$ is produced as a typical Gaussian k-Nearest-Neighbor similarity matrix (see section 2.2.2.3); the fundamental parameters set for this method are the followings.

- **Nearest neighbors**. The number of nearest neighbors is fixed to $k = 20$, this choice is due mainly according the average magnitude of the datasets employed and to avoid the generation of graph structure too complete, which can not depicts properly the essential heterogeneity of the training data.

- **Distance measure**. In general the more appropriated measure of distance depends of the kind of feature vectors. In our experiments we have always considered Euclidean distances (see section 2.2.2.1) since according the nature of the datasets employed is sufficiently suitable.

- **Kernel widths**. First of all, we denote with $\text{linspace}(a, b, n)$ and ideal function which returns a set of n linearly spaced real-valued numbers between the ends a and b (both included). For the kernel width space is established a specific window, that can be expressed as the following function

$$\begin{aligned} \phi(\delta) &= \text{linspace}(0.1\delta, \delta, 5) \cup \text{linspace}(\delta, 10\delta, 5) \\ &= \{\sigma_1, \sigma_2, \ldots, \sigma_9\} \end{aligned}$$

  which returns a finite subset of nine kernel widths according the input value δ (we remind that consists to the average distance of each points with respect to its k nearest neighbors). You can observe that would be possible to produce a different weighed graph for each kernel width available; anyway in our experiments we decided to choose that built with the greatest value, i.e. σ_9 (according the increasing order of the kernel widths), since contains clearly sharper similarity signals by the Gaussian filtering (see section 2.2.4.1).

In our work we consider experiments made on symmetric graph only, therefore after the construction of the Gaussian k-NN weighted adjacency matrix $\mathbf{W}$, we perform also a symmetrization based on the *maximum similarity* approach (see section 2.2.4.2). Moreover we recall that indirectly GTG algorithm applies by itself a Laplacian normalization of the input graph (see sections 5.4). Therefore putting together all these aspects the effective learning structure is summarized as a symmetric and Laplacian normalized Gaussian 20-Nearest-Neighbor graph, based on Euclidean distances among feature vectors; since it is employed a Gaussian kernel all the measures of similarity treated (the graph edge weighs) are defined in $[0, 1]$.

6.3 Validation approaches

6.3.1 Crisp misclassification

We design a specif evaluation strategy to guarantee good quality and reliability of the collected results, which is based mainly on classification error rate according the predictions returned by GTG algorithm (see section 2.4.5).

Given a generic dataset $D = \{(\mathbf{f}_1, y_1), (\mathbf{f}_2, y_2), \ldots, (\mathbf{f}_n, y_n)\}$ of n objects over a domain of m labels $\mathcal{Y}$, we may prepare a training set for semi-supervised learning (see section 2.1.2) simply deciding that subset of points from D is considered labeled. Formally we denoting with $D^{(train)} = (D_\ell, D_{-\ell})$ the training set where $D_\ell \subset D$ and $D_{-\ell} = \{\mathbf{f} \mid (\mathbf{f}, y) \in D \setminus D_\ell\}$ is the reaming part of D but without the known labels. Therefore we may see the Graph Transduction Game as a model

$$\mathcal{GTG}_{D^{(train)}} : D_{-\ell} \to \mathcal{Y}$$

which depends from the specif training set.

To get the evaluation faithful we establish some constrains to define the subdivision between labeled and unlabeled points. First of all it is necessary that holds $|D_\ell| \ll |D_{-\ell}|$, therefore we could model a maximum ratio such that $|D_\ell| < \alpha n$ (with $\alpha \in [0, 1]$) in order to hold such condition.[1] The second fundamental aspect to consider is about what support to give for each category in $\mathcal{Y}$ from the labeled points and the effect that such choice takes to learning process. Remaining the cluster assumption on the data and the graph transduction principle, we may express this observation: if the data tend to spread in separated clusters and the information of the labeled nodes is propagated to nodes in their neighborhoods, how will be estimated the labels for those unlabeled points which are located in cluster with no labeled points? In fact since we decide to select labeled points randomly there could occur that the supervised information for some classes is cut off from the learning, i.e. does not exist labeled points for a class. To reduce this uncertain situation for each category we guarantee that in D_ℓ there exists at least one point belonged to that;[2] another solution would be to follow the trivial rule where the number of labeled point is uniform for class (see section 6.4.5), but we have left this way since the cluster dimensions may be unbalanced in the dataset and very often this setting can not be practicable in real usages.

[1] Experimentally the choice of α is difficult to generalize for any datasets, but in our experiments we have decided empirically to remain in the window $\alpha \in [0.02, 0.1]$, which may be considered as a suitable tradeoff.

[2] Clearly this approach does not guarantee always that all the clusters are covered by at least a labeled point since some category can be described by multiple well-separated data groups; anyway we assume that such scenario occurs sporadically and hence negligible.

Graph Transduction Game is a semi-supervised model that performs transductive inference over the data, which has to be evaluated in a manner lightly different than the typical inductive models (see section 2.1.2). In the latter in fact the main principle consists to learn once and then predicting over unknown data; therefore to emulate this aspect in terms of evaluation it is sufficient that the test cases are completely external with respect to the training set. Instead as concern the validation on transductive learning, there occurs the test samples are completing part of training set $D^{(train)}$, which are determined just from the selection of unlabeled objects as $D^{(test)} = D \setminus D_\ell$ (in other terms training and prediction phases may be seen as an unique macroprocess).

To avoid measure of error ratio which depends from a specific data division it is convenient to produce different splits over the same dataset D and compute an average measure of performance. Moreover for a proper analysis the measure should focus a specific condition for the data split, which is its number of labeled points. Formally, denoting with $\mathcal{S}_D^{(l)}$ the finite set of several random selected data split $\left(D^{(train)}, D^{(test)}\right) \in \mathcal{S}_D^{(l)}$ whose the number of labeled objects in the training set is fixed to $l = |D_\ell|$, the average performance measure is computed as followings

$$\frac{1}{|\mathcal{S}_D^{(l)}|} \left(\sum_{(D^{(train)}, D^{(test)}) \in \mathcal{S}_D^{(l)}} \frac{|\{(\mathbf{f}, y) \in D^{(test)} \mid y \neq \mathcal{GTG}_{D^{(train)}}(\mathbf{f})\}|}{|D^{(test)}|} \right) \tag{6.1}$$

This validation strategy may be seen as a version of the more general repeated holdout approach (see section 2.4.5), which is adjusted for semi-supervised learning with transductive models.

In much parts of our experiments we compute multiple performance measures according an ideal sequence $l = |\mathcal{Y}|, \ldots, \approx \alpha n$ for the number of labeled points and for each choice l we performs 100 different data splits (i.e. $|\mathcal{S}_D^{(l)}| = 100$). This bag of data is finally depicted in a plot of average classification error rate per number of labels points with standard deviation ranges along the curves (see section 2.4.4).

6.3.2 Soft misclassification

The general misclassification performance which is considered in section 6.3.1 detects an error when a predicted label $\hat{y}$ for an instance $\mathbf{f}$ is wrong with respect to its real label y. Anyway this crisp principle does not verify how much the mistaken label is similar to the correct category. For example, if the set of labels is $\{orange, mandarin, dog\}$ and the object to classify is an orange, the error *mandarin* with respect to *dog* has a relevance indeed

different. Therefore we are interested to employ an evaluation technique that can take in account of the error extent, whose weight follows some measure of semantic/object based category similarity, in order to evaluate deeply the classifier [13, 51]. We may design two new forms of misclassification performance which employ of category similarities as followings:

- k-**Nearest Soft Misclassification**. Formally given a label $\lambda \in \mathcal{Y}$ we may model the set $L^k_\lambda \subset \mathcal{Y}$ that contains the first k labels which are more similar with respect to λ (included) according a measure of category similarity $s(\cdot)$. Therefore the rule to count an error during the evaluation is updated as the condition $\hat{y} \notin L^k_y$. The validation scheme as (6.1) is updated as followings

$$\frac{1}{|\mathcal{S}^{(l)}_D|}\left(\sum_{(D^{(train)},D^{(test)})\in\mathcal{S}^{(l)}_D}\frac{|\{(\mathbf{f},y)\in D^{(test)} \mid \mathcal{GTG}_{D^{(train)}}(\mathbf{f})\notin L^k_y\}|}{|D^{(test)}|}\right) \quad (6.2)$$

  In all the experiments which employ this form of performance the parameter k is set to 3, since we retain suitable that the wrong predictions fall at most within the first two different labels much more similar than the correct one. You can observe that this evaluation criterion is a sort of generalized scheme, since with $k = 1$ the rule (6.2) becomes identical to the method (6.1), which is based on the common misclassification error.

- **Free Soft Misclassification**. The main disadvantage of the measure based on k-nearest soft misclassification is the requirement of a fixed threshold, which cut off the real level of similarity between correct and wrong labels. Therefore the evaluation may be relaxed considering directly the cumulative amount of similarity weights, under the assumption to work with a measure of similarity $s(\cdot)$ defined in $[0, 1]$. This new scheme may be seen as a further more expressive generalization of the classical crisp misclassification. Considering the pair of labels for a certain test instance as $y, \hat{y} \in \mathcal{Y}$, in the latter the correct prediction is evaluated when $y = \hat{y}$ and counted as 1; exploiting of similarity instead for the same condition the result of the evaluation does not change, since equally $s(y, \hat{y}) = 1$. The unique difference is in the case of error $y \neq \hat{y}$, because may be considered a weight which is not always null as occurs in the crisp performance. Therefore the evaluation scheme as (6.1) may be updated as followings

$$\frac{1}{|\mathcal{S}^{(l)}_D|}\left(\sum_{(D^{(train)},D^{(test)})\in\mathcal{S}^{(l)}_D}\frac{\sum_{(\mathbf{f},y)\in D^{(test)}} s\Big(y, \mathcal{GTG}_{D^{(train)}}(\mathbf{f})\Big)}{|D^{(test)}|}\right) \quad (6.3)$$

6.4 Data analysis and experiments

In this section we introduce concretely whole survey made on Graph Transduction Game. First to focus in detail this topic we prefer to give a briefly description regarding the original experimental story of Graph Transduction Game [2], in order to have an overview more complete about all its known abilities with respect to the restrictive case of study in our dissertation.

Considering the initial aspect of the source of data, the evaluation tests were performed in several kind of datasets and feature descriptors too. Moreover the graph construction did not always follow our setting since were executed different deeper experiments, from which we summarize these particular differences:

- the distance measures employed were both euclidean and cosine based, clearly the distinction was according the type of objects in the dataset;
- for each dataset were inferred different graphs (one for kernel width available), but among the several performances collected was chosen the best only;
- the algorithm was tested both symmetric and asymmetric similarities;
- the definition of the kernel width space was revised for experiments with negative similarities (dissimilarity information).

Moreover the results of these particular tests were compared with other graph transduction algorithms well-known in machine learning community, in particular we cite the spectral graph transducer [52], the guassian fields and harmonic function-based method [53], the local and global consistency method [54], the Laplacian regularized least squared [55], the mixed label propagation method [56] and manifold regularization with dissimilarity method [57]. The conclusions derived from of all these experiments have proofed that Graph Transduction Game is very competitive with other solutions, being able in different cases to overcome them.

Although GTG has been already tested under different conditions in our survey we analyze in detail further several aspects, in order to understand more clearly how is featured its learning process; all our analysis is projected to reason about the strength relationship between data similarity and classification performances.

6.4.1 Preliminary dataset analysis

We start our survey making a brief analysis of the datasets (see section 6.1) employed to understand how the several instances are distributed and make

hypothesis of data similarity in spatial terms. For this aim we use a typical visual approach producing a scatter plot (see section 2.4.3) in bi-dimensional space using the t-Distributed Stochastic Neighbor Embedding [23] method to reduce data dimensionality; the membership for each class is distinguished by a proper color.

In figure 6.1 we have a complete depiction of the dataset Scene, whose own 8 categories $\mathcal{Y} = \{$*coast*, *forest*, *highway*, *inside city*, *mountain*, *open country*, *street*, *tall building*$\}$ seem quite ordered in terms of clustering subdivision.

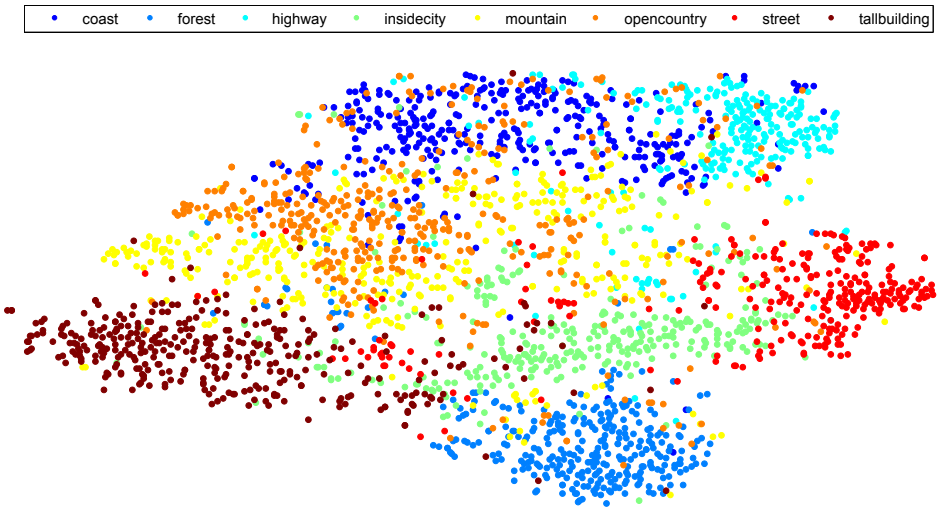


Figure 6.1: Scatter plot of Scene dataset with categories

Anyway looking more in detail we observe more overlapping mainly for the class *open country* and *mountain* up to *coast* and *forest*. From a semantic point of view we are not surprised of this fact, since the scenes of these classes are intuitively similar. In the same way we found weak correlation for the categories *tall building*, *inside city* and *street* too (although this fact is rather in conflict than a semantic interpretation of such classes).

The original datasets SUN and Caltech are too huge to be entirely analyzed and used in practical experiments, therefore we decide to work on smaller portions of data taking a reduced selection of categories and points from the complete data sources. We distinguish such two subsets with another nomenclature to avoid misunderstanding.

SUN-C8 dataset is built following almost the same structure of the Scene dataset: in detail we make a selection of 8 categories $\mathcal{Y} = \{$*creek*, *mountain*, *playground*, *restaurant*, *shopfront*, *living room*, *cockpit*, *cathedral (outdoor)*$\}$ with a fixed number of 350 scenes for a total of 2800 samples. In figure 6.2

we can see the peculiarities of our dataset which seem quite similar as Scene, but with lower level of separation for some categories; moreover for each category the images contained have a greater distance among themselves, i.e. they share less features, which suggest clusters much more condensed.

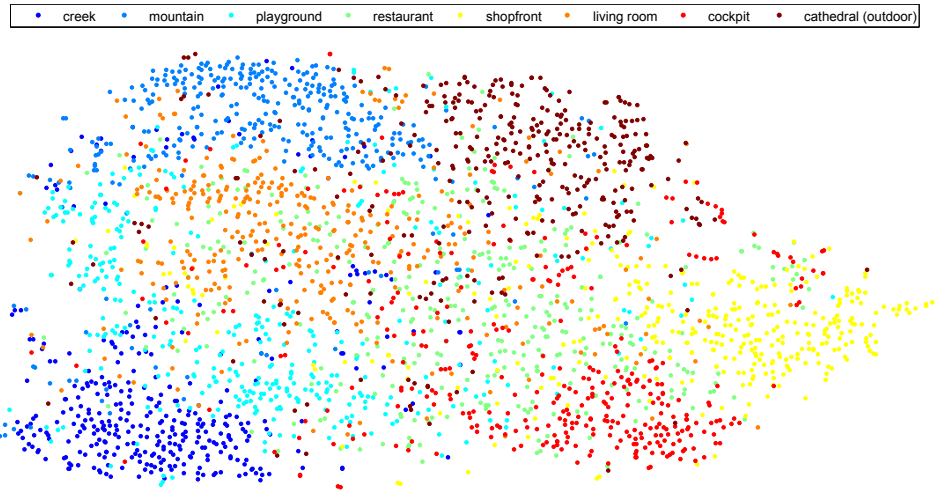


Figure 6.2: Scatter plot of SUN-C8 dataset with categories

It is interesting to discover that the categories *living room* and *restaurant* are particularly overlapped between themselves and surrounded by all the other ones. Considering the semantic of the real world this fact is not particularly clear since they refer to senses completely different; anyway we already know that in general the class of scene images is particularly inclined to arise this sort of results (as observed in the analysis of the similarity matrices in section 6.4.1).

Caltech-C6 dataset is built over a selection of 911 images of objects, organized in 6 different classes $\mathcal{Y} = \{$*bonsai*, *car side*, *chandelier*, *ketch*, *leopards*, *watch*$\}$, wherein for each one is given a mean support of 152 images.

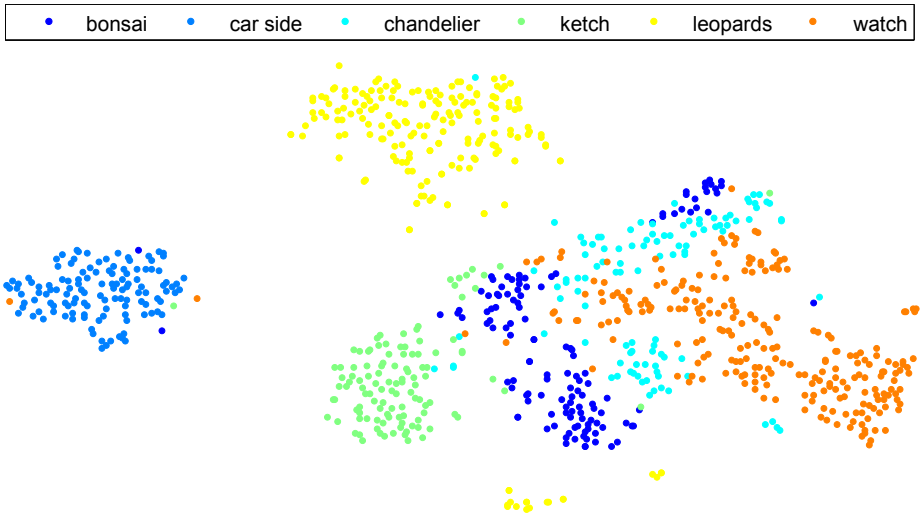


Figure 6.3: Scatter plot of Caltech-C6 dataset with categories

The figure 6.3 shows a situation indeed different with respect to Scene and SUN-C8, since the categories form well-separated clusters. Clearly the datasets are less comparable for several reasons, mainly for the different nature of the images and the number of classes. Moreover in this dataset is rather evident an interesting aspect bound to the data, in particular we observe that some categories, such as *leopards*, *chandelier* and *bonsai* may be formed by different well-separated groups; this fact is just in agreement with the fundamental definition of cluster assumption (see section 2.1.2).

6.4.2 Data similarity effect on the learning

In this part of our analysis we start eventually to familiarize with Graph Transduction Game through real usages over our datasets. Moreover we are interested to make a preliminary observation of its behavior to understand how the similarities among objects may influence the learning performances. For all the experiments the construction of the weighted graph is according the setting introduced in the section 6.2, while the evaluation of the results is based on typical crisp misclassification in section 6.3.1.

The first experiment is performed over the whole dataset Scene whose result are collected in the plot 6.4.

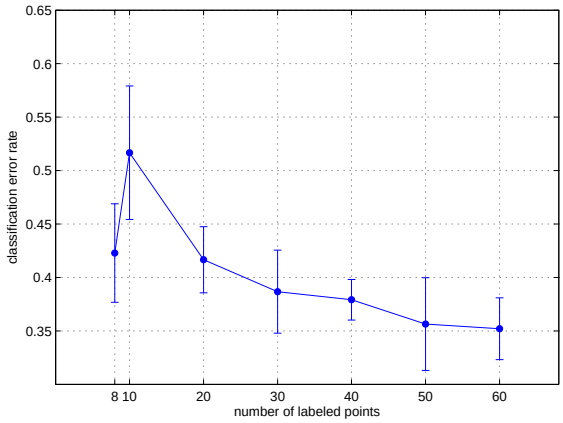


Figure 6.4: General GTG performance on Scene dataset

This trend suggests clearly that the support of labeled points is crucial for the performances of GTG (as widely treated in section 6.4.5), but also that the transductive learning works well in Scene dataset, since just employing 60 labeled points is possible to classify correctly more of 65% of all the 2688 scenes (about 1747 images).

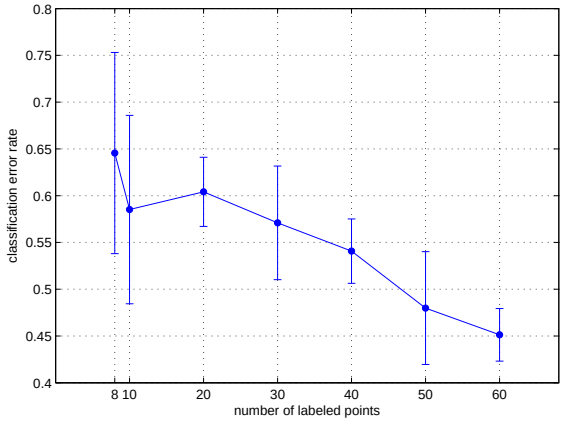


Figure 6.5: General GTG performance on SUN-C8 dataset

In the plot 6.5 we see the behavior of our second experiment with SUN-C8 dataset, which are quite worse with respect to the experiment 6.4. We remains that the construction of this dataset is made following the structure of Scene, in such way to obtain results more comparable. Therefore if the performance decreases have to be there some measurable reason; we could guess different and combined aspect that may explain this behavior, for example:

- the scenes inside the same category share less features, i.e. images of

the same category are visually low similar among themselves;

- the images belonged to different classes share many features, i.e. the categories are similar in visual terms;
- there is noise in the data (or outlier points), which may be seen as low quality images or scenes whose supervised label is unsuitable.

In fact if we compare the heatmaps of the graphs built for Scene and SUN-C8 datasets looking at the figure 6.7 in the latter the amount of external images which are similar is indeed greater, while those belonged to the categories are rather dissimilar (as it is visible also in the depictions 6.1 and 6.2); in other terms the cluster assumption over the dataset SUN-C8 is weaker then Scene, which involves negatively to the general performances.

The third experiment is performed over the dataset Caltech-C6, which contrary to Scene and SUN-C8 it contains images of objects.

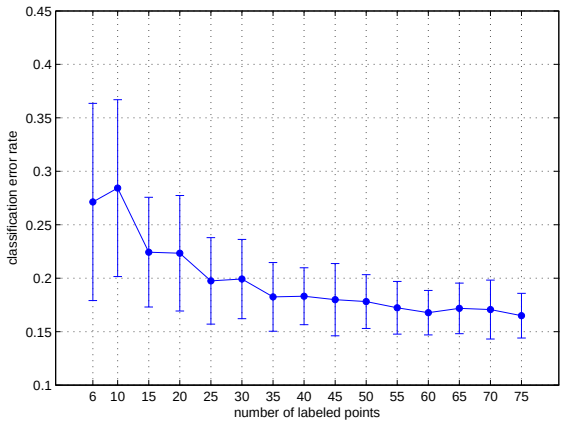


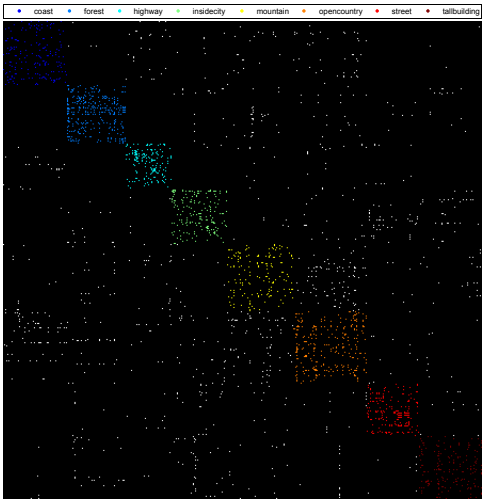
Figure 6.6: General GTG performance on Caltech-C6 dataset

The performances in plot 6.6 are really good already from small values of labeled points, which suggest a low degree of ambiguities to classify the data. Although it is important to consider that in Caltech-C6 the problem space with respect to our previous experiences is indeed reduced both in terms of number of classes and instances; therefore also this aspect has contributed to obtain better results.

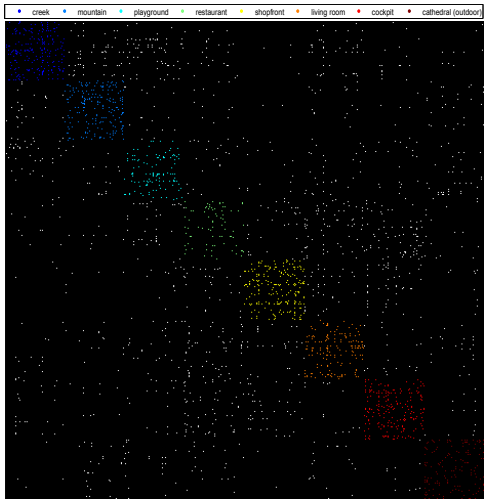
In the figure 6.7 as below we collect the heat maps of the weighed adjacency matrices built for all the three experiments; the several categories are distinguished with different colors. Looking these weights seems that the problem of similar images among different classes it is a trend much more visible for the scene than object pictures. This observation is not only an isolated case related to our datasets, but actually refers to a notorious

problem that arises just between these two kinds of pictures. In visual terms in fact is quite intuitive that any ambient figures which could be categorized in different classes may contain common artifacts, for example categories such as *open country*, *hill*, *forest*, *mountain* share vegetation, sky, animals and other elements. Therefore we do not wonder about this indeed gap of performances registered among Scene and SUN-C8 with respect to Caltech-C6. Clearly this aspect may occur also with objects images, especially when contains backgrounds, but experimentally has been observed to have a minor incidence.[1]

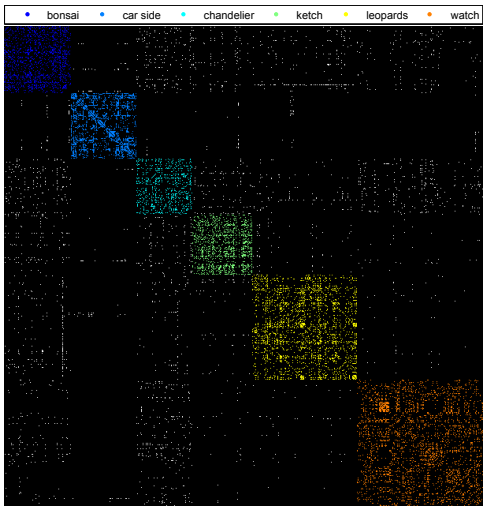
[1]To deal with this weakness about the scene images, in Computer vision there exist other interesting variant with respect to well-known GIST descriptor (see section 2.3.2.1); in that we cite for example the work [58] on SUN dataset, which is based on a representation with 102 high-level attributes. Anyway in this dissertation we are not interested to evaluate performance with respect to different types of feature vectors.



(a) Scene



(b) SUN-C8



(c) Caltech-C6

Figure 6.7: Similarity matrix/Gaussian 20-NN graph heatmaps for several datasets

The level of goodness of the results collected in these several experiments is supported by the measure of intra-class membership (see section 2.4.2) in the table 6.1 too, that proofs to us how the strength of the cluster assumption involves the learning process.

Scene	*SUN-C8*	*Caltech-C6*
63.66	52.79	79.67

Table 6.1: Levels of intra-class membership ratio (%) for each experiments

6.4.3 Category similarity adjustments

In our experimental analysis we use similarity measurements over the set of categories which are inferred from data objects or the WordNet ontology (see section 2.2.3). The original definition for some metrics do not give suitable measures of similarity which should be symmetric and defined in $[0, 1]$ (see normalization and symmetrization respectively in sections 2.2.4.3 and 2.2.4.2). These two properties are crucial for our survey, especially for the computation of soft misclassification (see section 6.3.2), the comparison between different measures and the experimental phase of GTGwCS (see section 6.4.7). Therefore we decide to specify once all the common adjustments we always made before to use these measures in our experiments.

- **Semantic-based category similarities**. For semantic-based category similarities (see section 2.2.3.1) we start to adjust the class names since some original labels in our dataset are unknown in WordNet. To solve this problem we have chosen a proper replacement as proposed in table 6.2.

Scene	SUN-C8	Caltech-C6
$insidecity \rightarrow city$	$cathedral(outdoor) \rightarrow cathedral$	$carside \rightarrow car$
$opencountry \rightarrow country$		$leopards \rightarrow leopard$
$tallbuilding \rightarrow skyscraper$		

Table 6.2: Labeling adjustments per dataset

  The word acceptation for each label is that considered most common, which in WordNet is specified by the first synset (sense number #1) where the category is contained (see section 2.2.3.1). Moreover for *Resnik*, *Jiang-Conrath* and *Lin* semantic similarity we use the British National Corpus. The measure that are unbalanced and defined in a general domain $\mathbb{R}^+$ are *Leacock-Chodorow* and *Resnik* similarities; for this reason we apply normalization and symmetrization (with maximum criterion) for both.

- **Object-based category similarities**. For object-based category similarities (see section 2.2.3.2) we start often from a random selection $D' \subseteq D$ of instances which are established according the requirements of the analysis to perform. In detail for the measures based on aggregated similarities according the principles of *minimum*, *mean* and *maximum*, we can either skip the choice of D' to exploit directly of any preexisting normalized similarity matrix (e.g. based on the weighted adjacency matrix $\mathbf{W}$ used by GTG) or choose a subset D' where inferring a similarity

matrix from a precomputed Euclidean-based distance matrix (see section 2.2.2.1). The measures of *centroid divergence* and *visual distance* have to be always computed from a subset D' according the operational steps described in 2.2.3.2. For those measures which involve a choice of an initial subset D' is always required a halfway conversion from a distance measure to a normalized similarity in $[0, 1]$, which follows in our case the common rule (2.4); in order to avoid to work on saturated measures, the scaling parameter chosen in this phase is always $\gamma = 8$. At this step all the measures are defined in $[0, 1]$ but some could not be balanced, in particular for *minimum*, *mean* and *visual distance* category similarities, where we apply normalization and symmetrization (with maximum criterion) for the last two only. Unfortunately we can not adjust *minimum* object-based category similarity since the maximum level is not centered (i.e. $\exists \lambda \in \mathcal{Y} : \max_{\mu \in \mathcal{Y}}(s_{\lambda\mu}) \neq s_{\lambda\lambda}$); anyway we solve the problem with a non conventional rule simply forcing that for each label $\lambda \in \mathcal{Y}$ holds $\hat{s}_{\lambda\lambda} = 1$ (we consider this trick an acceptable tradeoff for the nature of the measure and our aims) .

All the other measures which are not cited are already normalized and symmetric for definition.

6.4.4 Category analysis of the errors

The misclassification ratio considered in section 6.4.2 gives an overall measure of performance without to say nothing about the effective entities of the wrong predictions. We are questioning if there exist a correlation between the mistaken labels with respect to the correct classes, in other terms whether objects from similar categories tends to be more confused.

In order to verify this hypothesis we compute an object-base category similarity matrix $\mathbf{S}^{\mathcal{V}} = (s^{\mathcal{V}}_{ij})$ with average affinity criterion (see section 2.2.3.2) from the visual similarities $\mathbf{W}$ which describe the graph built for the dataset Scene (as specified in 6.2). That we should obtain is a shoot about the similarities of the categories in visual terms seen by GTG (since the process knows just $\mathbf{W}$ about the data). The table 6.3 is a simply way to show briefly part of the information in $\mathbf{S}^{\mathcal{V}}$ where for each category we list the first three similar categories ordered for visual similarity level (from the more similar to less).

Category	$+ \leftarrow$ Visual similarity $\rightarrow -$			
coast	*opencountry*	*highway*	*mountain*	...
forest	*mountain*	*tallbuilding*	*insidecity*	...
highway	*coast*	*opencountry*	*mountain*	...
insidecity	*street*	*tallbuilding*	*opencountry*	...
mountain	*opencountry*	*forest*	*tallbuilding*	...
opencountry	*mountain*	*coast*	*highway*	...
street	*insidecity*	*highway*	*mountain*	...
tallbuilding	*insidecity*	*forest*	*mountain*	...

Table 6.3: Category per similar categories of Scene ordered for visual similarity

It is important not to fall in the error to expect a sharp matching with our human knowledge of the real world, i.e. to see a correct order with respect to the semantic senses of the categories. In fact the results in $\mathbf{S}^{\mathcal{V}}$ reflect the semantic in Scene dataset, which is another story; for example, the second similar class *highway* with respect to *coast* in the first row, it would be more suitable if was placed at least after *mountain* (making a general semantic interpretation), but it is evident that in Scene the images of *highway* share much more aspects than those of *mountain* in relation to *coast*.

Nevertheless we perform another interesting analysis to proof that the similarities according $\mathbf{S}^{\mathcal{V}}$ are correlated to the wrong predictions made by GTG. In the plot 6.8 we show a distribution of the errors collected during all the 100 runs of GTG. If we denote as usual the set of all 8 classes as $\mathcal{Y}$, each category $\lambda \in \mathcal{Y}$ in the x-axis denotes a correct supervised label. The distribution built over λ shows the degree of those erroneous classes whose the label λ is been wrongly predicted; for example, we can see that the images of classes *opencountry* are confused in majority with the label *mountain*, then *coast* and so on.

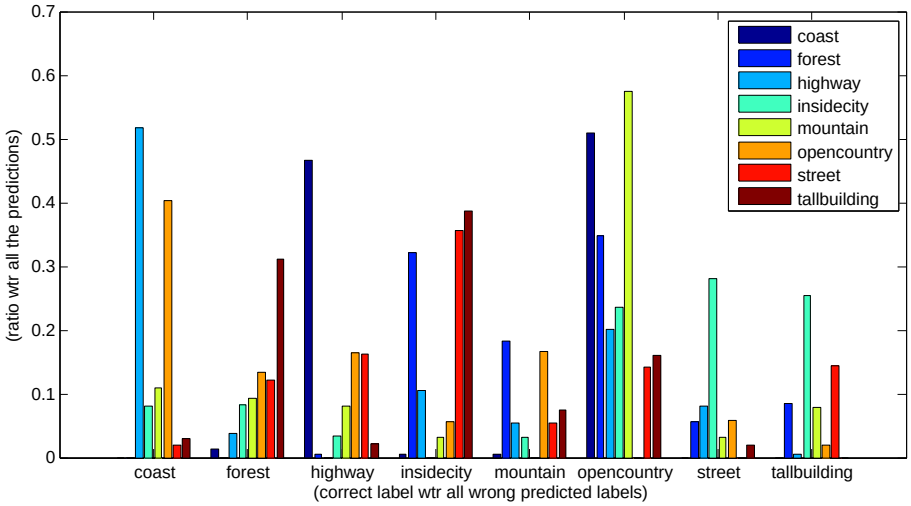
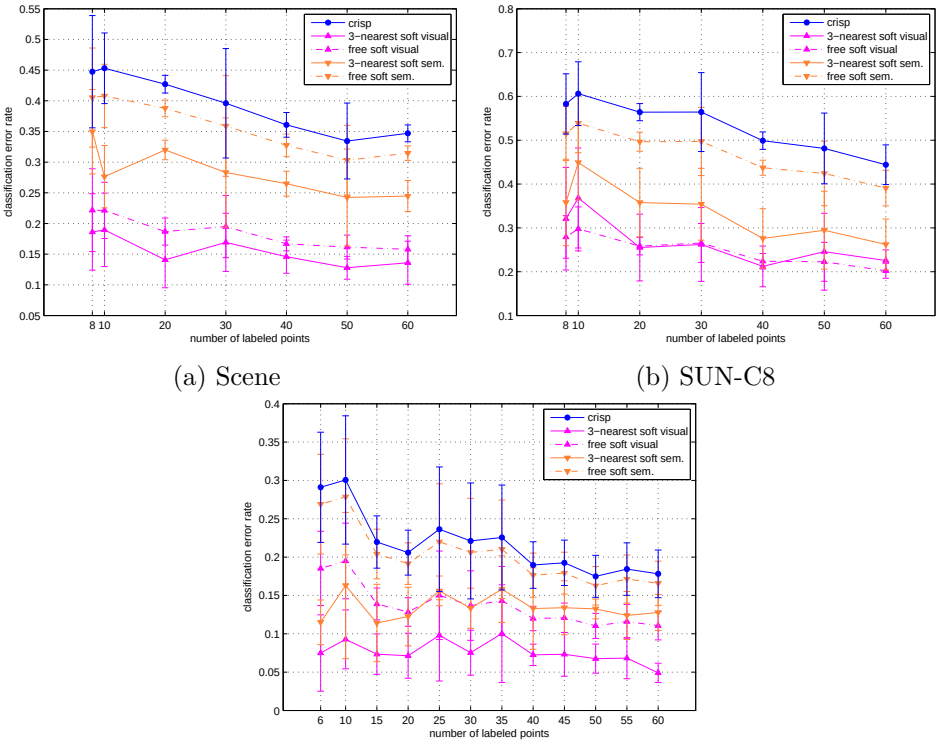


Figure 6.8: Error distribution of correct class per unmatched labels from several trials on Scene dataset

At this point, making a comparison between the table 6.3 and the plot 6.8 we can observe keeping a reference class λ that the labels more mistaken are just those more similar to λ (at least for the majority of the cases). This experimental analysis suggests that our hypothesis is true, i.e. according the category similarity of the dataset the wrong predictions are those more similar to their correct labels. For this reason the similarity among different classes has a relevant effect for the general performances.

Another way to verify the same aspect consists to introduce category similarity to evaluate the results through the soft misclassification approach (see section 6.3.2). In the figure 6.9 we compare both crisp and the two forms of soft performances with nearest-based category threshold and free soft misclassification for all our datasets; moreover the category similarities taken in account are both visual and semantic, respectively obtained by mean criterion (see section 2.2.3.2) and shortest path measure (see section 2.2.3.1).



(a) Scene (b) SUN-C8

(c) Caltech-C6

Figure 6.9: Crisp and Soft performances on several datasets

We can observe that the results based on soft misclassification are better in general than crisp misclassification, in particular way for the 3-nearest soft misclassification; another interesting aspect consists to notice how visual similarity catches much more affinity among the real outcomes and wrong predictions with respect to the semantic measure. This panorama is a further proof that the greater part of the errors are among similar categories. Moreover it is important to observe that this evaluation method may be employed for other extended forms of classification as for example multi-label learning [59], i.e. multiple categories for a single objects; this new topic remains to a hidden structure in GTG, the Nash equilibrium mixed strategy profile, which intrinsically already contains multiple object-based estimations of more likely memberships on the data, which are clearly lost by maximum *a posteriori* rule (5.3). Anyway we remark that our experimental study is funded on *ex post* observation of the Graph Transduction Game.

We conclude our analysis of the errors showing a particular depiction inspired to the visual representation of a dataset as in section 6.4.1. Our idea consists to see where are placed the wrong instances in the data space.

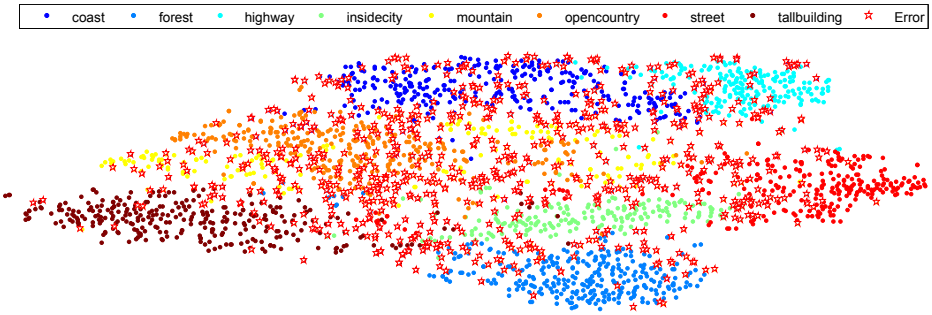


Figure 6.10: Scatter plot of Scene dataset with must frequent wrong objects

The figure 6.10 shows the usual scatter plot of the dataset Scene where we mark the objects that are mistaken most frequently, i.e. the instances that have been harder to classify correctly according all the runs of GTG (in this way we can present a result which is quite independent with respect to a specific selection of labeled nodes in the learning). Comparing the plain figure 6.1 with 6.10 we can see that the most part of the errors are placed in the regions of overlapping of several categories. This fact proofs again the problem of the visual similarity sharing of the categories, which reduces the cluster assumption principle confusing the transductive learning process.

6.4.5 Choosing labeled objects

A crucial condition that involves on the graph transduction learning is the choice of the labeling points. We are interested to study the behavior of GTG about the combination of these following aspects:

- *those* labeled points has been chosen;
- *how many* labeled points has been chosen.

We start our analysis focusing on a single category $\lambda \in \mathcal{Y}$ of Scene dataset. In detail we want to see if the amount of the errors increases when the chosen labeled points are in the neighbourhood of overlapping points from different categories (with respect to λ). The transductive learning in graph transduction approach propagates the information from labeled points to those unlabeled whose share similarity, or in other terms all the nodes which are reachable in the graph from them. The associated graph $\mathcal{G}$ of Scene is depicted by a weighed adjacency matrix $\mathbf{W}$ which is symmetric, therefore $\mathcal{G}$ is an undirected graph. A category may be formed as more separated clusters, in other terms by several connected components. Although if we consider a single connected component that contains a point of label λ it is not guarantee that all the other reachable points share the same label (since there exist edges among objects which are similar in visual terms, but from other categories). These stranger points are considerable as a sort of noise

that dirty the cohesion of the category depicted by a connected component. We decide to collect these structures building a subgraph $\mathcal{G}^{(\lambda)}$ of $\mathcal{G}$ which is the union of all the connected components that contain at least a node of category λ. In the scatter plot 6.11 we show the images associated to the nodes of $\mathcal{G}^{(\lambda)}$ for all the eight classes in Scene, where the black dots denote just the reachable nodes of stranger classes inside the same connected components of λ.

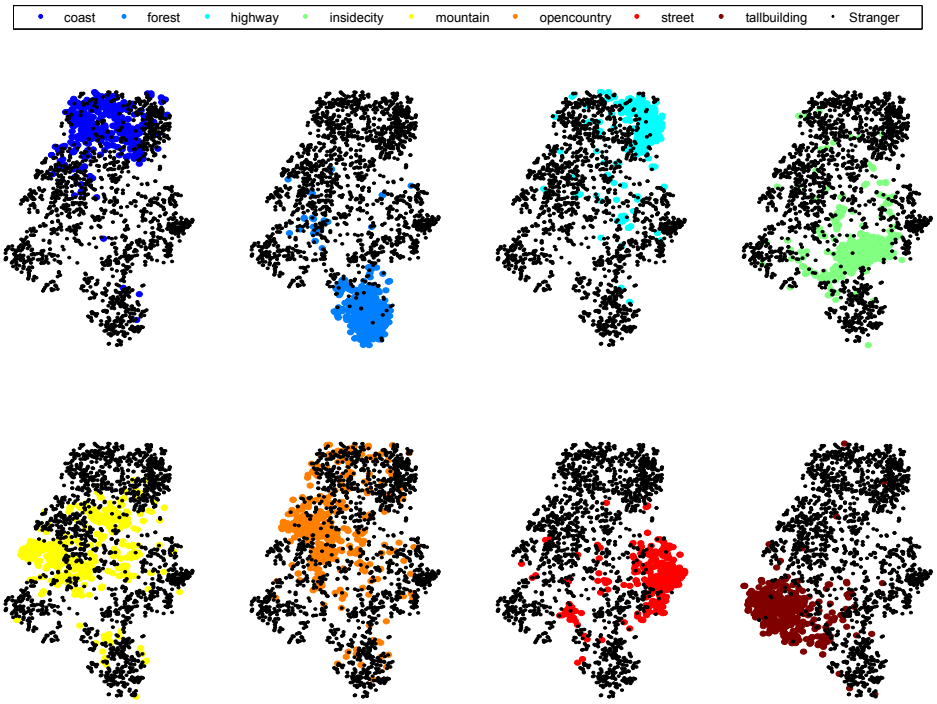


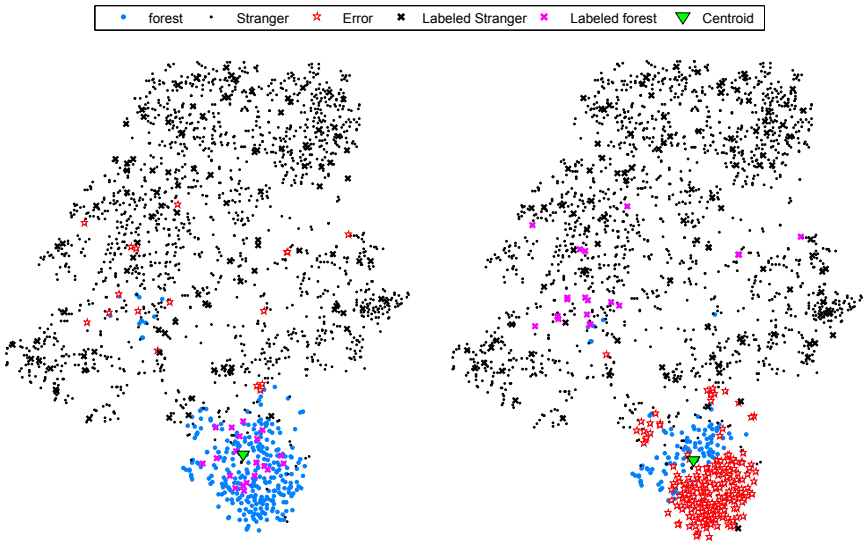
Figure 6.11: Connected component for class with stranger nodes of Scene dataset[1]

This depiction is indeed interesting since we can see that there exists a high degree of stranger points connected to a single category, which reveals how similarity between objects may introduce relationships which does not reflect the real supervised categorization.

After this overall vision of the graph $\mathcal{G}$ we decide to continue our analysis working with the class *forest* only since it seems quite well-defined (i.e. it is presumably formed by an unique predominant connected component). The experiment consists to execute two runs of GTG selecting 20 labeled points per class but in such way those of *forest* are either near (a) or far away (b)

[1]The stranger nodes which are integrated in the representation are detected considering edges with similarity weights over 60%.

with respect to the neighborhood detected by the category centroid.



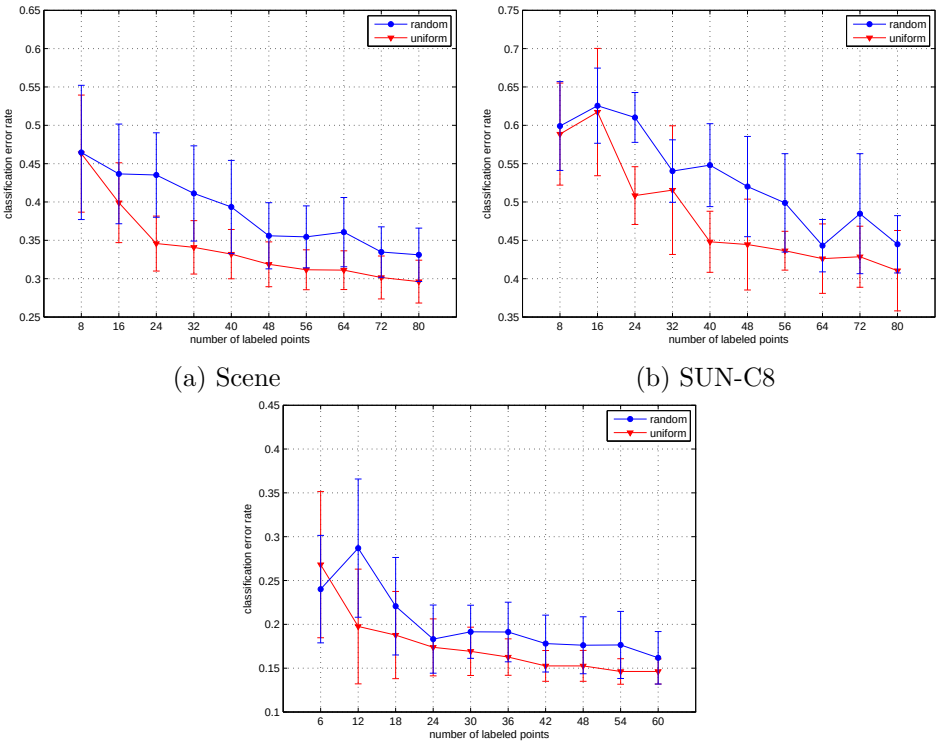
(a) Labeled points near to centroid (b) Labeled points far from centroid

Figure 6.12: Two learning experiment for class *forest* with respect to different labeled points

In the plot 6.12 are represented all the labeled points and wrong predictions especially related for the class *forest*. This result is indeed interesting since shows how is crucial the choice of labeling points in the learning. In general the category membership of any labeled points is propagated along the reachable neighbor nodes, but if they are integrated in connected component which would represent better other classes, then will generate confusion about the predictions of the unlabeled points. In the case (a) the labeled points of class *forest* falls in the principal region which describes that same class, therefore the cluster receive a strong contribute for the correct label. In the case (b) instead since the support from labeled nodes of class *forest* is confined in far and dispersive regions with respect to the reference cluster of this category, then their contribute will be too weak to solve all the unlabeled nodes of class *forest*, which will be classify mainly by the wrong suggestions given from those stranger labeled nodes which are nearer in the main area.

The amount of labeled points clearly involves on the general performances since it reduces the size of the test set, i.e. the number of errors that is possible to commit. Anyway we may perform a study considering the increasing of the labeling points with respect to how they are selected. In detail we are questioning if for a given amount of labeled points the performances are different if they are selected randomly or equally distributed for all the categories. The crisp misclassification method introduced in section 6.3.1

is based just to select a certain number of labeled points in random order over the whole dataset, we may adjust this principle dividing their amount with the same proportion for each class (the selection of labeled points is still random, but inside the same category only).



(a) Scene

(b) SUN-C8

(c) Caltech-C6

Figure 6.13: Performances with labeled points randomly selected and uniformly per class on several datasets

In plot 6.13 we make a comparison of the classical approach with respect our new idea for our datasets; to read these new performances is sufficient to consider that the amount in the x-axes l means that each class receives exactly $\frac{l}{|\mathcal{Y}|}$ labeled points. The results are interesting since the second approach works better already from small selections. The reason may be motivated implicitly from that we have understood in the previous analysis of the graph structure. In fact if we select labeled points for each class is more likely they cover much more connected components that forms such category, we could say that the supervised knowledge is democratically distributed (since for each class in the datasets employed there is available approximately the same number of objects)[1]; instead with a random selection can be given

[1]It is right wondering if a dataset which contains some categories much more supported than others involves negatively to the learning performances. Anyway from our experience

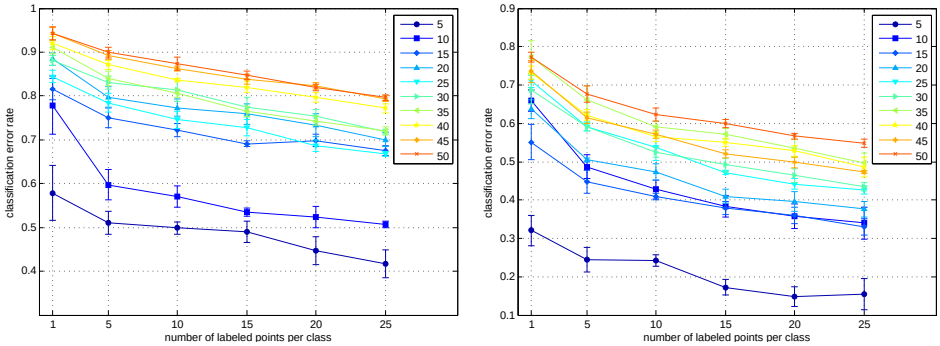
more help for certain classes than other ones. However gradually the number of labeled points is increasing the method based on random selection will converge to the uniform criterion, since it is less likely that some classes is more preferred.

6.4.6 Scalability over multiple classes

In the section 6.4.5 we study the effect of the labeled points on the performances without consider the amount of classes involved. We have to remind that GTG is mainly a multi-class classifier, therefore it is necessary to study how the number of classes affects the learning; in other terms how much Graph Transduction Game is class scalable.

For this analysis we employ the complete datasets $D = \text{SUN-397}$ and $D = \text{Caltech-101}$ introduced in section 6.1, since we can rely on a very huge selection of images and categories $\mathcal{Y}$. In detail, for an experiment with a subset $\mathcal{Y}^{(k)} \subset \mathcal{Y}$ of k different classes randomly selected, we build a dataset $D^{(k)} \subset D$ extracting for each class $\lambda \in \mathcal{Y}^{(k)}$ a fixed number z of objects; finally over the dataset $D^{(k)}$ we compute as usual the crisp misclassification performances. For a comparative analysis we prefer to merge the results given for multiple experiments $k = k_1, k_2, \ldots, k_n$, but to obtain a faithful observation we should allow that in the learning of any datasets is given the same help in terms of labeled points, i.e. they scale with the number of classes. According the evaluation strategy in section 6.3.2, we can not control properly the distribution of the labeled points for each $D^{(k_1)}, D^{(k_2)}, \ldots, D^{(k_n)}$. Therefore we redefine the evaluation strategy just computing performance on a fixed number of labeled points $q \ll z$ per number of classes k_i which is shared for all $i = 1 \ldots n$, in other terms for any dataset $D^{(k_i)}$ the real number of labeled points chosen for the learning depends only from k_i and it is obtained as $l_{D^{(k_i)}} = qk_i$ (in this way for different choices of q the learning is always democratic for all the experiments). In detail we perform $n = 10$ experiments where for each $i = 1 \ldots 10$ the number of classes is $k_i = 5i$; moreover we make six selections of labeled points per class starting from a base case $q_0 = 1$ to the others $j = 1 \ldots 5$ cases as $q_j = j5$ (for each class of a dataset $D^{(k_i)}$ the q_j labeled points are randomly selected). Finally in the construction of the dataset in all the experiments the classes receive a support of $z = 100$ different images, hence $|D^{(k_i)}| = 100k_i$.

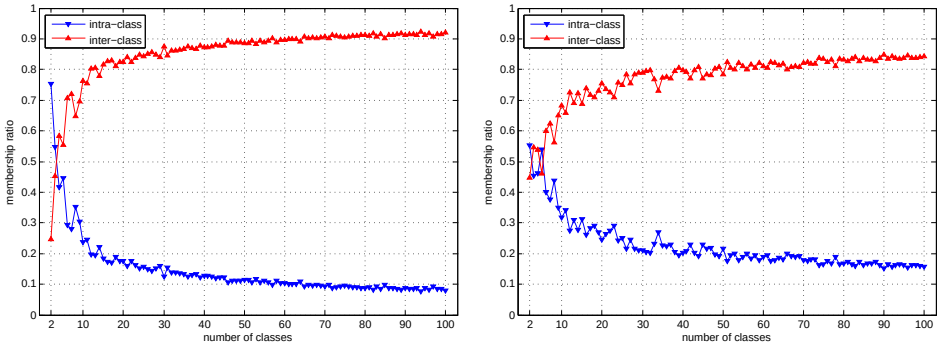
we consider this aspect negligible since it is much more relevant the similarities among objects than their magnitude, unless this degree of imbalance is indeed heavy.



(a) SUN-397 (b) Caltech-101

Figure 6.14: Performances per different values of classes on several datasets

In figure 6.14 we can observe that already over 10 classes the performances registered are rather weak and the trend worsens much more quickly with higher values of classes. It is important to remark the real fact that working on many classes increasing indeed the category similarity sharing and in general degrade the cluster assumption on the data. In the figure 6.15 we proof in detail this fact, showing simply with 20 instances per category how the level of intra-class membership ratio (see section 2.4.2) decreases with respect to the number of classes and the symmetric inter-class measure underlines the reinforcement of the similarity degree between objects of different categories. Therefore this should be a typical context where category similarity could help to go beyond such limit.



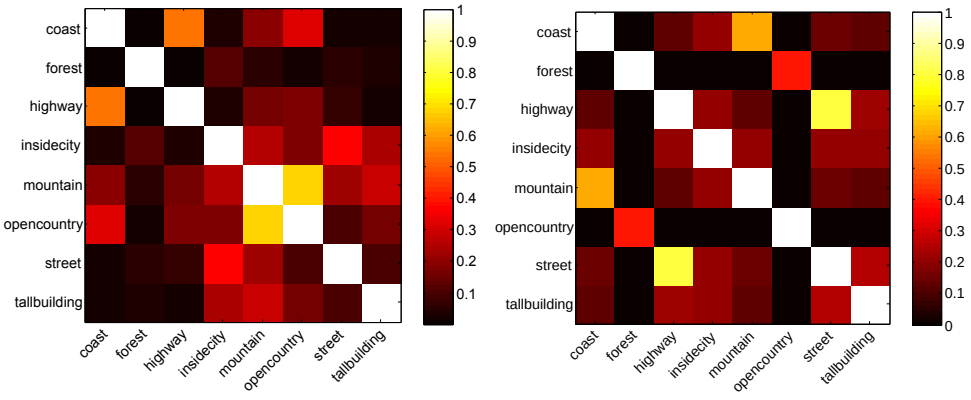
(a) SUN-397 (b) Caltech-101

Figure 6.15: Intra/Inter-class membership ratio per number of classes on several datasets

With this analysis we discover that GTG is not particularly scalable in terms of data heterogeneity; although this fact does not have to underestimate GTG, simply we may say that is not adapted to work with wide scales of classes.

6.4.7 Learning with category similarity

The category similarities establish a precious high-level information which may be used for different tasks. So far we have used them to evaluate in deep the behavior of GTG, in the general setting which exploits of the visual similarity among the objects only. It is presumable that introducing class similarities in the learning process should help to decrease ambiguities and as consequence increasing the performances. An important advantage of (semi)supervised learning consists to have whole knowledge about the involved category domain $\mathcal{Y}$ for all the objects. This condition allows to get possible the computation of the label similarities among all the classes even before to start the training phase. In the section 2.2.3 we introduced two main approaches to infer category similarities, where the former is based on the linguistic semantic extracted from the sense network WordNet and the latter on the observation of the similarities among objects with respect to their membership. From a conceptual point of view it is licit to wonder if the similarity relation described from these two source of data are comparable and in particular similar. We are questioning if the category similarity from a dataset is another facet of the high level semantic category similarity, where we consider the latter as the nearest formalization of the human knowledge. We already know that exists in general correlation between visual and semantic similarity [14], but not how much these two kinds of information can be interchangeable for real usages. In the category analysis of the errors in section 6.4.4 we have already observe how object-based category similarities can be not particularly understandable in semantic terms, but we could make a direct comparison with effective semantic-based category similarities over the same classes. For such test we employ the dataset Scene to compute a visual-based category similarity matrix $\mathbf{S}^{\mathcal{V}} = (s^{\mathcal{V}}_{\lambda\mu})$ with centroid divergence method (see section 2.2.3.2); for the same classes, we compute a semantic category similarity matrix $\mathbf{S}^{\mathcal{T}} = (s^{\mathcal{T}}_{\lambda\mu})$ using the Jiang-Conrath measure (see section 2.2.3.1), which is well-considered for several linguistic analysis [60].



(a) $\mathbf{S}^{\mathcal{V}}$ heatmap (b) $\mathbf{S}^{\mathcal{T}}$ heatmap

Figure 6.16: Object and Semantic based category similarities from Scene dataset

In the figure 6.16 we see in detail all the category similarity levels as heat maps (see section 2.4). It is immediately visible how $\mathbf{S}^{\mathcal{V}}$ and $\mathbf{S}^{\mathcal{T}}$ are dissimilar, i.e. they not describe the same relations for the domain of classes, at least for the greater part of configurations. In the table 6.4 this fact is even more evident, where we collect as usual for each class and method the first three classes ordered for the similarities.

Category	$+ \leftarrow$ Visual similarity $\rightarrow -$				$+ \leftarrow$ Semantic similarity $\rightarrow -$			
coast	*highway*	*opencountry*	*mountain*	...	*mountain*	*insidecity*	*street*	...
forest	*insidecity*	*street*	*mountain*	...	*opencountry*	*insidecity*	*street*	...
highway	*coast*	*opencountry*	*mountain*	...	*street*	*insidecity*	*mountain*	...
insidecity	*street*	*mountain*	*tallbuilding*	...	*street*	*mountain*	*opencountry*	...
mountain	*opencountry*	*tallbuilding*	*insidecity*	...	*coast*	*insidecity*	*street*	...
opencountry	*mountain*	*coast*	*highway*	...	*forest*	*insidecity*	*street*	...
street	*insidecity*	*mountain*	*opencountry*	...	*highway*	*insidecity*	*mountain*	...
tallbuilding	*mountain*	*insidecity*	*opencountry*	...	*insidecity*	*street*	*highway*	...

Table 6.4: Category per similar categories of Scene ordered for visual and semantic category similarity

We made these comparisons also for other measures of similarity, but the general answer derived from our heuristic observation suggests that is preferable to consider visual and semantic category similarity as two different separated sources of information, since in real usages there are too aspects that involve on their distance. Although it is interesting to decide that of these two forms of similarities would be more suitable in the learning.

The visual category similarities $\mathbf{S}^{\mathcal{V}} = (s^{\mathcal{V}}_{\lambda\mu})$ from a dataset D may been seen as a membership digest from the similarities among all the objects in D; for this reason the visual category similarity $s^{\mathcal{V}}_{\lambda\mu}$ is always correlated to the visual similarity between two objects w_{ij} in D whose classes are

respectively $y_i = \lambda$ and $y_j = \mu$. Therefore the information in $\mathbf{S}^{\mathcal{V}}$ may act as an augmentative factor for the contextual hypothesis made for the objects. In a certain senses it should help the learning to continue for a course that implicitly already known. The semantic category similarities $\mathbf{S}^{\mathcal{T}}$ are clearly a new additional information which is totally external with respect to the working dataset. Therefore it should be able to give a contribute even more distinctive than that object-based, since these measures may unweighted misleading hypothesis supported by visually similar objects which belong to different categories actually.

It is important to observe that the computation of visual-based category similarity matrix $\mathbf{S}^{\mathcal{V}}$ clearly depends mainly from the origin dataset, while the information in the semantic category similarity matrix $\mathbf{S}^{\mathcal{T}}$ is global (considering a fixed reference sense network), i.e. depends explicitly by $\mathcal{Y}$ only and may be reused for any datasets categorized over the same domain of labels. Although the objects in the training set may be hand labeled according a category acceptation which may be particularly far with respect to the common sense of the words; if this peculiarity of the dataset was not known, namely supervised, then it could be better to employ object-based category similarity.

On the basis of all the considerations as above, we may say that does not exist an unique answer about the choice between visual or semantic category similarities in the learning, but the latter is surely preferable in the majority of the cases since it should depict better the human knowledge.

In the section 5.5 we guess a way to introduce category similarity in the Graph Transduction Game, the *Graph Transduction Game with Category Similarity* (GTGwCS), which is formulated simply as a generalization of the compatibility matrix modeled in the original insight. We start to explore concretely our GTGwCS making before some experimental observation in order to understand how to control this new form of learning; finally we use GTGwCS in real cases to study respectively the performances obtained with visual and semantic category similarity. In all the experiments the reference evaluation approach continue to be based on crisp misclassification (see section 6.3.1), but with the new extended conceptual model expressible as

$$\mathcal{GTG}_{\left(D^{(train)},\mathbf{S}^{(\mathcal{Y})}\right)} : D_{-\ell} \rightarrow \mathcal{Y}$$

6.4.7.1 Preliminary dynamical observations of GTGwCS

Introducing new additional information in the learning consists in a novel approach of GTG. Therefore we prefer to observe firstly what happens simulating a real learning under toy category similarities, in other terms we

want to check the stability of GTGwCS while the structure $\mathbf{S}$ progressively goes away from the base case $\mathbf{I}_{|\mathcal{Y}|}$. We design this analysis with several rules to compute all the elements of a general, normalized and symmetric matrix $\mathbf{S}_t = \Big(s_t(\lambda,\mu)\Big)$ (since we work in real cases with symmetric category similarity measures normalized in $[0,1]$); moreover for each experiment $t = 1, 2 \dots n$ the off-diagonal elements are randomly determined in a domain scaled according a fixed parameter $a \in \mathbb{R}^+$ as followings.

(a) **Variable random selection**. Choosing random values in the window $[0, at]$ such that:

$$s_t(\lambda,\mu) = s_t(\mu,\lambda) = \begin{cases} s_t \in [0,at] & \lambda \neq \mu \\ 1 & \text{otherwise} \end{cases}$$

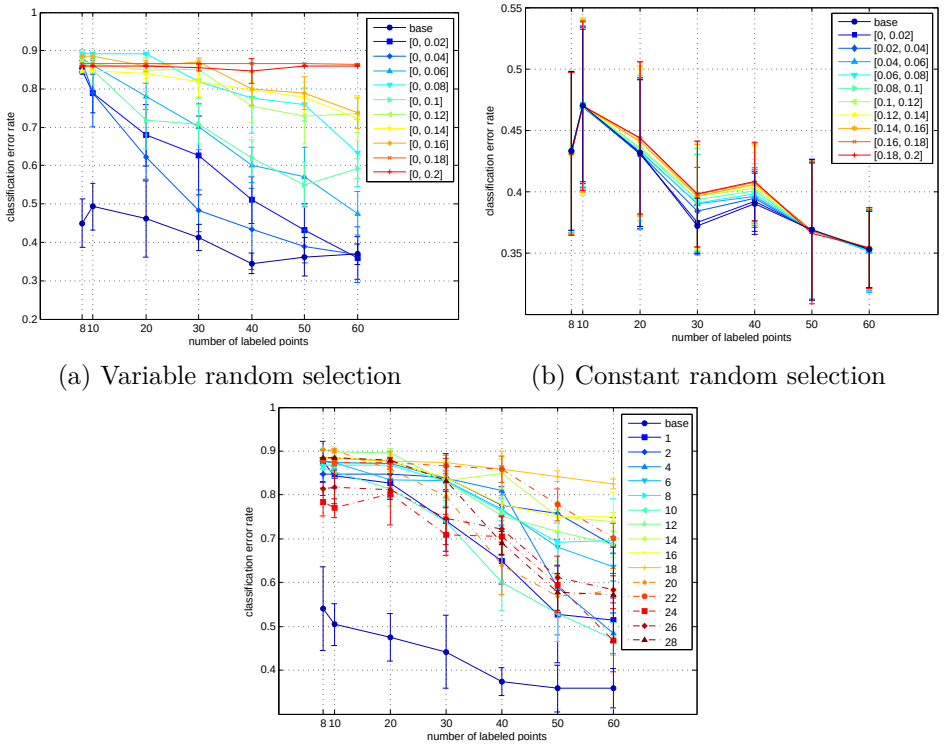
(b) **Constant random selection**. Choosing a constant random value in the window $s_t \in [a(t-1), at]$ such that:

$$s_t(\lambda,\mu) = s_t(\mu,\lambda) = \begin{cases} s_t & \lambda \neq \mu \\ 1 & \text{otherwise} \end{cases}$$

(c) **Incremental Random filling**. Established a sub set of t non trivial configurations $P_t \subset \mathcal{Y}^2$ (i.e. $\forall(\lambda,\mu) \in P_t,\ \nexists(\alpha,\beta) \in P_t : (\alpha,\beta) = (\mu,\lambda)$ and $|P_t| = t$), choosing random values in the window $[0,a]$ such that:

$$s_t(\lambda,\mu) = s_t(\mu,\lambda) = \begin{cases} s_t \in [0,a] & (\lambda,\mu) \in P_t \\ 1 & \lambda = \mu \\ 0 & \text{otherwise} \end{cases}$$

All the perturbation tests are made on Scene dataset computing as usual classification performances (see section 6.3.1) where: for both the methods (a) and (b) we decide to perform until $n = 5$ experiments over the fixed parameter $a = 0.02$; for the method (c) we perform experiments for several choices until to fill whole structure with $n = \binom{|\mathcal{Y}|}{2} = 28$ (the binomial coefficient of $|\mathcal{Y}|$=8 choose 2), over the fixed parameter $a = 0.1$. Clearly to understand the extent of the variations we add the reference performance where simply we assign $\mathbf{S}_0 = \mathbf{I}_{|\mathcal{Y}|}$. Moreover the selection of labeled objects for each different experiment t is always the same for a much more accurate comparison (as formally explained in section 6.4.7.3 too).



(a) Variable random selection

(b) Constant random selection

(c) Incremental Random filling

Figure 6.17: Perturbation performance analysis with GTGwCS on Scene dataset

In figure 6.17 we see the final results of all our experiments where we can underline several aspects. The case (a) shows how GTGwCS is sensible already from small values of category similarities; this fact suggests that the information contained in $\mathbf{S}_t$ has to be adequately adjusted for real usages (see section 6.4.7.3). In the analysis (b) instead the trend is stable, this behavior is quite understandable since the bag of perturbation is democratically distributed in $\mathbf{S}_t$. Finally the analysis (c) shows how the learning process is sensible already with little information available in $\mathbf{S}_t$.

6.4.7.2 Category similarity adjustments for the learning

The tips introduced in 6.4.3 allows to prepare any our measure of similarity in terms of normal form $[0, 1]$ and symmetric property. This decision is important since it limits the study with category similarity measures that contain the base case $\mathbf{I}_{|\mathcal{Y}|}$, which we know is fundamental to model the transductive learning of GTG (see sections 5.2 and 5.3). Anyway according the experimental observations in section 6.4.7.1 we can not introduce this

information directly in the learning since the process seems really sensitive. Therefore in order to avoid possible predominance whit respect to the object-based information, we design a general rule which is applied before to start the learning on any category similarity measures $\mathbf{S}_t = \Big(s_t(\lambda, \mu)\Big)$ of type t (already normalized and symmetric) as followings

$$\hat{s}_t(\lambda, \mu) = \exp\Big(-\eta\Big(1 - s_t(\lambda, \mu)\Big)\Big) \tag{6.4}$$

where the positive parameter $\eta \in \mathbb{R}^+$ is used to manage the decay factor of the similarity level. The parameters are empirically learned for each experiments according the principle that the new performances obtained (i.e. after the introduction of $\mathbf{S}_t$) remains in a the window $[0, p_0 + \delta]$ with respect the first one registered for the base case p_0; the threshold of tolerance imposed is always $\delta = 0.2$.

6.4.7.3 Experiments with visual category similarities

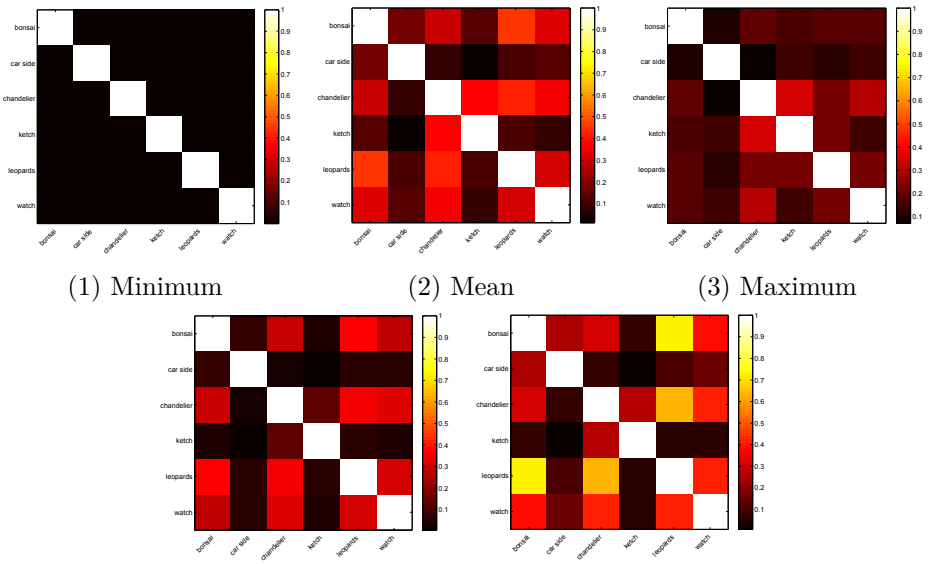
The measures of visual category similarities should be much more possible external with respect to effective training set used to built the graph structure $\mathcal{G}$ in GTG, but clearly always in harmony with the same nature of the objects to classify. In fact it is less reasonable to compute $\mathbf{S}^{\mathcal{V}}$ using the same objects to infer the related weighted adjacency matrix $\mathbf{W}$ for several aspects:

- The real information introduced in the learning is redundant, because implicitly already contained in the training set.
- The computation of visual category similarities requires whole supervised data, but due to semi-supervised learning we know only the labeling objects, which represent actually just a minimal part of all the instances to classify. Therefore the visual category similarities further to be redundant, they get invalid the learning performances since in a certain sense there would mean to suggest the correct labeling *a priori*.

According these conditions we design a new evaluation strategy. Formally for a generic dataset $D = \{(\mathbf{f}_1, y_1), (\mathbf{f}_2, y_2), \ldots, (\mathbf{f}_n, y_n)\}$ of n objects over a domain of labels $\mathcal{Y}$ we extract for each class a sufficient fixed number of objects (whose value is decided on the basis of available data in D) to build a new subset $D_{\mathcal{V}} \subset D$ taken in account to compute $\mathbf{S}^{\mathcal{V}}$ (according the tips in section 6.4.3); after this step the reaming part $D_{\mathcal{L}} = D \setminus D_{\mathcal{V}}$ is used to compute $\mathcal{G}$ for all the experiments with GTGwCS.

In the section 2.2.3.2 we introduce five methods to compute the matrix $\mathbf{S}_t^{\mathcal{V}}$, therefore we are interesting to see the performances obtained for each measure $t = 1 \ldots 5$. Moreover to understand the distance with respect to

the original formulation of GTG, we add the performance without visual category similarities, denoting as usual with $t = 0$ the general base case $\mathbf{S}_0^{\mathcal{Y}} = \mathbf{I}_{|\mathcal{Y}|}$. However to get faithful the comparison among all the category measures, the selection of labeled and unlabeled points in $D_{\mathcal{L}}$ have to be maintained in all the measure. Formally (according the evaluation method in section 6.3.1) let $\mathcal{S}_{D_{\mathcal{L}}}^{(l)}$ the finite set of random selected data split of l labeled objects, we compute the crisp misclassification performances sharing the unique fixed division $\left(D_{\mathcal{L}}^{(train)}, D_{\mathcal{L}}^{(test)}\right) \in \mathcal{S}_{D_{\mathcal{L}}}^{(l)}$ for each visual category similarity $\mathbf{S}_t^{\mathcal{Y}}$ of type t.



(1) Minimum (2) Mean (3) Maximum

(4) Centroid divergence (5) Visual distance

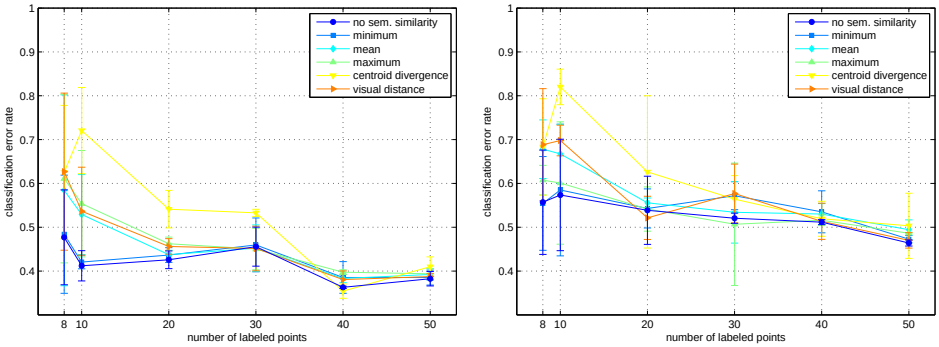
Figure 6.18: Heatmap per visual category similarity measures for Caltech-C6 dataset

In figure 6.18 we give a briefly panorama of the several visual category similarities computed for Caltech-C6 dataset; we can see the strong correlation among all the metrics, since they furnish with different weights information rather similar, in particular way clearly for the types $t = 2, 4, 5$.

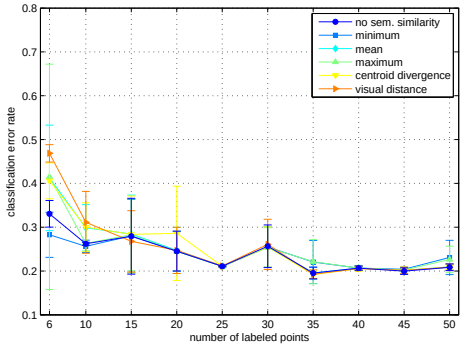
t	*Method*	Scene η_t	SUN-C8 η_t	Caltech-C6 η_t
1	minimum	1	1	1
2	mean	15	6	8
3	maximum	12	5	5
4	centroid divergence	12	6	15
5	visual distance	18	8	26

Table 6.5: Learned scaling parameters for visual category similarities per datasets

The figure 6.19 presents eventually all the performances obtained over our datasets according the learned experimental parameters in table 6.5. Making a preliminary analysis we do not observe relevant advantages about the introducing of visual category similarity on any measures. Anyway the worst results are got when the number of labeled points is low, while progressively all they converge to the normal GTG without category similarities.



(a) Scene (b) SUN-C8

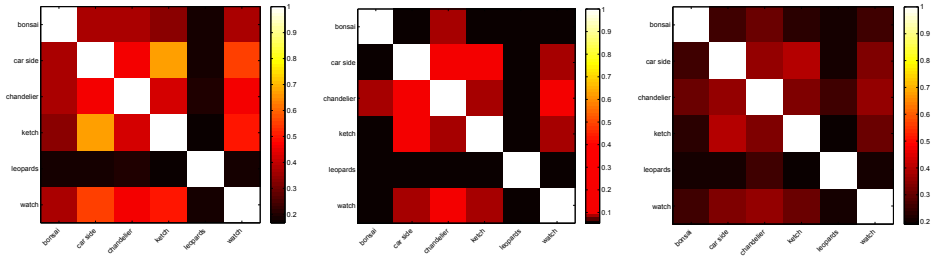


(c) Caltech-C6

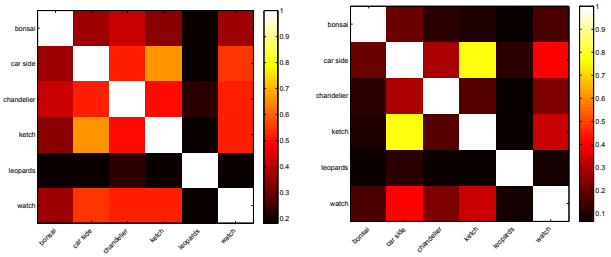
Figure 6.19: Performances with visual category similarity on several datasets

6.4.7.4 Experiments with semantic category similarities

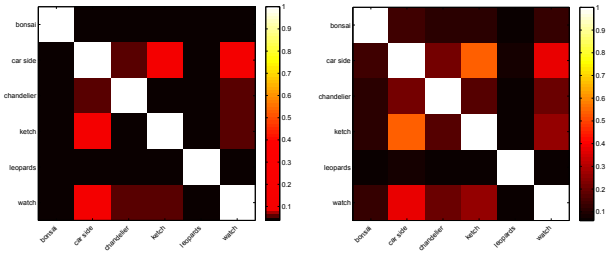
In the section 2.2.3.1 we introduce seven measures of semantic similarity which are exploited to compute the matrix $\mathbf{S}_t^{\mathcal{T}} = \Big(s_t(\lambda, \mu)\Big)$ according the text labels associated to the categories. The strategy of evaluation is as that decided for visual category similarity (see section 6.4.7.3), but clearly the learning can be simply performed over the complete dataset (in other terms $D_{\mathcal{L}} = D$) since $\mathbf{S}_t^{\mathcal{T}}$ is inferred from the sense network based of WordNet, therefore it can never be a redundant information with respect to the objects to classify (or the similarities in $\mathbf{W}$).



(1) Semantic Relatedness (2) Shortest Path (3) Leacock-Chodorow



(4) Wu-Palmer (5) Resnik



(6) Jiang-Conrath (7) Lin

Figure 6.20: Heatmap per semantic category similarity measures for Caltech-C6 dataset

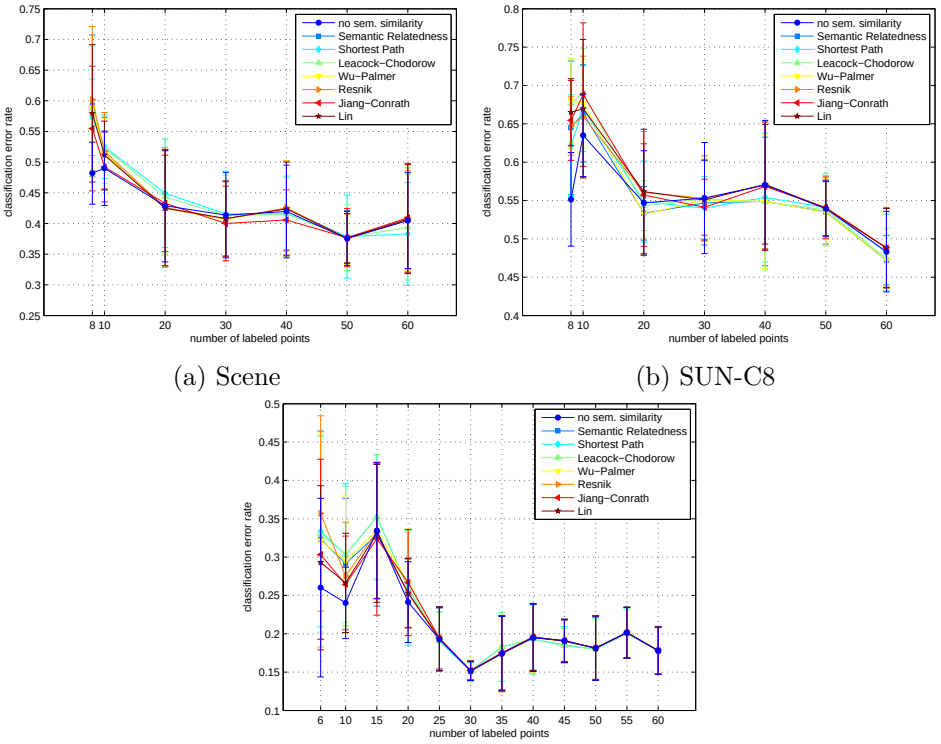
In figure 6.20 we introduce the several semantic similarity measures computed for Caltech-C6 dataset. As already observed with visual category similarities (see section 6.4.7.3), all the metrics share a common evaluation for the labels involved; another visible aspect is how semantic similarity

can distinguish sharper such configuration, in particular we see that class *leopards* and *bonsai* have less in common with respect to the semantics between themselves and all the other categories.

t	*Similarity*	Scene η_t	SUN-C8 η_t	Caltech-C6 η_t
1	Semantic Relatedness	23	13	21
2	Shortest Path	6	4	6
3	Leacock-Chodorow	13	8	12
4	Wu-Palmer	29	13	26
5	Resnik	26	19	12
6	Jiang-Conrath	6	4	5
7	Lin	20	10	11

Table 6.6: Learned scaling parameters for semantic category similarities per datasets

The figure 6.21 introduces as usual the performances obtained over all the datasets according the learned parameters in table 6.6. The behavior seems to be quite similar to the experiments with visual category similarity (see section 6.4.7.3), as concern the comparison with the base case and the trend on small numbers of labeled points. Therefore not even the best candidate information of semantic category similarity can improve the classification performance of GTG on our datasets.



(a) Scene

(b) SUN-C8

(c) Caltech-C6

Figure 6.21: Performances with semantic category similarity on several datasets

Making a general evaluation according all the registered results, we retain that very likely the real problem is not due to the type of category similarity measure but just to the theoretical model of GTGwCS. It is indeed interesting to realize how this process attempts to abort much more possible the information of category similarity given, in such way to take back GTGwCS toward the original schema of GTG; in other terms remaining anchored on the polymatrix game structure (see section 3.5.1) it seems to not exist a better solution to improve the general performances.

Reasoning in terms of the underlying relaxation labeling scheme (see section 4.3) the generalization of GTG seems quite suitable, since the contextual information is a combination of visual and category similarity. Anyway if we analyze in deep such model looking to the transductive learning principle devised in GTG (see section 5.2), we could guess a possible reason to justify this fact. The cluster assumption is a strong hypothesis made on the input data, but GTGwCS impoverishes this principle to generalize the case of different categorization for similar objects, equally to state that objects which are similar may have similar labels. Moreover it is difficult that an unique

common set of pairwise category similarities can be sufficiently reliable to represent the membership sharing for each local object in the dataset. It very likely that GTGwCS's scheme becomes too expressive/general in practice usages and hence chaotic.

Chapter 7

Conclusions

In this dissertation we explored several crucial aspects of semi-supervised learning through the novel *Graph Transduction Game* (GTG) [2]. Our first analysis was about the fundamental information of similarities among objects to understand those properties have to hold in order to support properly an inference task. We saw how the cluster assumption or homophily principle whose GTG devises its transductive learning represents the crucial condition for the application of such model. Over this preliminary observation we investigated other deeper aspects which involved the learning as labeled objects, multi-category scale and classification trends estimating these following conclusions.

- The number of labeled objects chosen affects on the performances, but actually it is only the consequence of another real cause, which is the degree of labeled instances spread uniformly in well separated clusters; in other terms, it is less important how much supervised information is given than the quality of itself.
- The greater part of objects which are misclassified falls in regions of the data space whose neighbor instances do not share the same membership, equally they are mistakenly predicted with categories which appears similar to other ones.
- The number of categories clearly contributes on the complexity of the model, but the reason which the classification accuracy decreases is due to the impoverishment of the cluster hypothesis, since much more objects of different classes tends to increase the similarity sharing among themselves.

The observation of these answers suggested us that the main problem of all the story consists that objects heterogeneously categorized not should share features, but clearly this scenario does not match with data in the real world.

In order to perform our survey we focused the specif context of computer vision, employed as instances of interest common images of scenes and objects. To evaluate the behaviors of GTG we exploited of another form of similarity, which is instead projected on the domain of categories and inferred directly on the visual data. In literature we discovered that visual similarity is correlated to another higher level information, the linguistic semantic behind the categories [14]. We learned that exist measures which may suggest *a priori* those categories are similar and in there it was obvious to wonder if this new knowledge may be used to reduce ambiguities in a learning task. As marginal study of our analysis we decided to test a possible approach introducing category similarity information in the original version of GTG, forecasting a generalized model which called *Graph Transduction Game with Category Similarity* (GTGwCS). The initial analysis of the dynamics in GTGwCS already suggested that the stability of the base case was too compromised. Anyway we continued our study proofing that both with visual and semantic category similarities the performances obtained were not significantly better with respect to the original GTG.

Enlarging this particular topic we found interesting correlated works wherein semantic similarity is employed as an extension for common classifiers [45] or concretely introduced in the learning as minimization problems [51, 59]. As concern GTG was clear that the constrain of cluster assumption it is indeed strength to be easily smoothed through our variant GTGwCS, which allows higher expressiveness for the category relations, too much maybe. Moreover we could guess that the evolutionary model [33] whose GTG is anchored very likely is compromised. In that we are supported by the observation of the well-known *quasispecies equation* [61]; GTGwCS in a certain sense seems to imitate such schema, which contrary to replicator function employed by GTG, it models the mutation dynamics where is open the possibility to generate new categories or pure strategies. In other terms the fundamental global Nash equilibrium related to whole population of categories is lost generating confusion that takes to unsuitable labeling assignments.

In this dissertation we made a wide overview about what involved semi-supervised learning on large-scale categorization. Graph Transduction Game has been our tool to analyze in deep this important field, which has revealed interesting ability over a moderate number of classes. The study on the implications taken by reality samples of the visual world and how these may be discriminated for classification task, there has allowed us to understand what is been made but how much is even necessary to do.

Bibliography

[1] H. W. Kuhn, *Lectures on the Theory of Games*, Princeton University Press, **2003**.

[2] A. Erdem, M. Pelillo, "Graph Transduction As a Non-cooperative Game" in Proceedings of the 8th International Conference on Graph-based Representations in Pattern Recognition, Springer-Verlag, Münster, Germany, **2011**, pp. 195–204.

[3] J. Nash, "Non-Cooperative Games", *The Annals of Mathematics - Second Series* **Sept. 1951**, *54*, 286–295.

[4] R. A. Hummel, S. W. Zucker, "On the Foundations of Relaxation Labeling Processes", *IEEE Trans. Pattern Anal. Mach. Intell.* **Mar. 1983**, *5*, 267–287.

[5] K. Marriott, P. J. Stuckey, *Programming with Constraints: An Introduction*, MIT Press, **1998**.

[6] E. Tsang, *Foundations of Constraint Satisfaction*, Academic Press, London and San Diego, **1993**.

[7] J. Han, M. Kamber, *Data Mining, Second Edition: Concepts and Techniques*, Morgan Kaufmann, 2nd, **2006**.

[8] A. R. Mehryar Mohri, A. Talwalkar, *Foundations of Machine Learning*, MIT Press, **2012**.

[9] O. Chapelle, B. Schlkopf, A. Zien, *Semi-Supervised Learning*, The MIT Press, 1st, **2010**.

[10] G. Teschl, *Ordinary Differential Equations and Dynamical Systems*, American Mathematical Society, 1st, **2012**.

[11] C. Fellbaum, *WordNet: An Electronic Lexical Database.* Cambridge, MA: The M.I.T. Press, **1998**.

[12] A. Budanitsky, G. Hirst, "Evaluating WordNet-based Measures of Lexical Semantic Relatedness", *Comput. Linguist.* **Mar. 2006**, *32*, 13–47.

[13] R. Fergus, H. Bernal, Y. Weiss, A. Torralba, "Semantic Label Sharing for Learning with Many Categories" in Proceedings of the 11th European Conference on Computer Vision: Part I, Springer-Verlag, Heraklion, Crete, Greece, **2010**, pp. 762–775.

[14] T. Deselaers, V. Ferrari, "Visual and Semantic Similarity in ImageNet" in Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, Washington, DC, USA, **2011**, pp. 1777–1784.

[15] A. Oliva, A. Torralba, "Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope", *Int. J. Comput. Vision* **May 2001**, *42*, 145–175.

[16] S. Lazebnik, C. Schmid, J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories" in Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2, IEEE Computer Society, Washington, DC, USA, **2006**, pp. 2169–2178.

[17] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", *Int. J. Comput. Vision* **Nov. 2004**, *60*, 91–110.

[18] P. Lingras, R. Yan, C. West, "Comparison of Conventional and Rough K-means Clustering" in Proceedings of the 9th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, Springer-Verlag, Chongqing, China, **2003**, pp. 130–137.

[19] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, Y. Gong, "Locality-constrained Linear Coding for image classification" in Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, **2010**, pp. 3360–3367.

[20] L. van der Maaten, E. O. Postma, H. J. van den Herik, "Dimensionality Reduction: A Comparative Review", *Journal of Machine Learning Research* **2009**, *10*, 1–41.

[21] I. Jolliffe, *Principal Component Analysis*, Springer, 2nd, **2002**.

[22] I. Borg, P. Groenen., *Modern Multidimensional Scaling: Theory and Applications*, Springer, 2nd, **2005**.

[23] L. van der Maaten, G. Hinton, "Visualizing Data using t-SNE", *Journal of Machine Learning Research* **Nov. 2008**, *9*, 2579–2605.

[24] M. Pavan, M. Pelillo, "A New Graph-theoretic Approach to Clustering and Segmentation" in Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, Madison, Wisconsin, **2003**, pp. 145–152.

[25] M. Pavan, M. Pelillo, "Dominant Sets and Hierarchical Clustering" in Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2, IEEE Computer Society, Washington, DC, USA, **2003**, p. 362.

[26] J. M. Smith, *Evolution and the Theory of Games*, Cambridge University Press, **1982**.

[27] C. Daskalakis, P. W. Goldberg, C. H. Papadimitriou, "The Complexity of Computing a Nash Equilibrium", *Commun. ACM* **Feb. 2009**, *52*, 89–97.

[28] G. van der Laan, A. J. J. Talman, L. van der Heyden, "Simplicial Variable Dimension Algorithms for Solving the Nonlinear Complementarity Problem on a Product of Unit Simplices Using a General Labelling", *Math. Oper. Res.* **Aug. 1987**, *12*, 377–397.

[29] S. Govindan, R. Wilson, "A global Newton method to compute Nash equilibria", *Journal of Economic Theory* **2003**, *110*, 65 –86.

[30] T. R. Kaplan, J. Dickhaut, "A Program for Finding Nash Equilibria", *Math. J.* **1991**, *1*, 87–63.

[31] O. L. Mangasarian, "Equilibrium Points of Bimatrix Games", *Journal of the Society for Industrial and Applied Mathematics* **1964**, *12*, 778–780.

[32] R. Porter, E. Nudelman, Y. Shoham, "Simple Search Methods for Finding a Nash Equilibrium" in Games and Economic Behavior, **2008**, pp. 642–662.

[33] J. Weibull, *Evolutionary Game Theory*, Cambridge, MA: The M.I.T. Press, **1995**.

[34] J. Hofbauer, K. Sigmund, *Evolutionary Games and Population Dynamics*, Cambridge University Press, **1998**.

[35] J. T. Howson, "Equilibria of Polymatrix Games", *Management Science* **1972**, *18*, 312–318.

[36] R. M. Haralick, L. G. Shapiro, "The Consistent Labeling Problem: Part I", *IEEE Trans. Pattern Anal. Mach. Intell.* **Feb. 1979**, *1*, 173–184.

[37] A. Rosenfeld, R. A. Hummel, S. Zucker, "Scene Labeling by Relaxation Operations", *Systems Man and Cybernetics IEEE Transactions on* **1976**, *SMC-6*, 420–433.

[38] R. M. Haralick, L. S. Davis, A. Rosenfeld, D. L. Milgram, "Reduction operations for constraint satisfaction", *Information Sciences* **1978**, *14*, 199 –219.

[39] T. Elfving, J.-O. Eklundh, "Some properties of stochastic labeling procedures", *Computer Graphics and Image Processing* **1982**, *20*, 158–170.

[40] M. Pelillo, "The Dynamics of Nonlinear Relaxation Labeling Processes", *J. Math. Imaging Vis.* **Oct. 1997**, *7*, 309–323.

[41] D. A. Miller, S. W. Zucker, "Copositive-plus Lemke Algorithm Solves Polymatrix Games", *Oper. Res. Lett.* **July 1991**, *10*, 285–290.

[42] E. David, K. Jon, *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*, Cambridge University Press, New York, NY, USA, **2010**.

[43] S. Rota Bulò, I. M. Bomze, "Infection and immunization: A new class of evolutionary game dynamics", *Games and Economic Behavior* **2011**, *71*, 193–211.

[44] G. Zappella, "A Scalable Multiclass Algorithm for Node Classification", *CoRR* **2011**, *abs/1112.4344*.

[45] R. Fergus, H. Bernal, Y. Weiss, A. Torralba, "Semantic Label Sharing for Learning with Many Categories" in Proceedings of the 11th European Conference on Computer Vision: Part I, Springer-Verlag, Heraklion, Crete, Greece, **2010**, pp. 762–775.

[46] R. J. Sternberg, *Cognitive psychology*, Australia Thomson/Wadsworth, 5th ed., **2009**.

[47] G. Lakoff, *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*, University Of Chicago Press, **1990**.

[48] E. Rosch, "Cognitive Representations of Semantic Categories", *Journal of Experimental Psychology* **1975**, *104*, 192–233.

[49] J. Xiao, J. Hays, K. Ehinger, A. Oliva, A. Torralba, "SUN Database: Large-scale Scene Recognition from Abbey to Zoo" in Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, **2010**.

[50] L. Fei-Fei, R. Fergus, P. Perona, "Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories", *Comput. Vis. Image Underst.* **Apr. 2007**, *106*, 59–70.

[51] B. Zhao, L. Fei-Fei, E. Xing, "Large-Scale Category Structure Aware Image Categorization" in Proceedings of the Neural Information Processing Systems (NIPS), **2011**.

[52] T. Joachims, "Transductive Learning via Spectral Graph Partitioning" in Proceedings of the 20th Intl. Conf. on Machine Learning, Cambridge, MA: MIT Press, **2003**, pp. 290–297.

[53] X. Zhu, Z. Ghahramani, J. Lafferty, "Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions" in Proceedings of the 20th Intl. Conf. on Machine Learning, Cambridge, MA: MIT Press, **2003**, pp. 912–919.

[54] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, B. Schölkopf, "Learning with local and global consistency" in Advances in Neural Information Processing Systems 16, MIT Press, **2004**, pp. 321–328.

[55] M. Belkin, P. Niyogi, V. Sindhwani, "Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples", *J. Mach. Learn. Res.* **Dec. 2006**, *7*, 2399–2434.

[56] W. Tong, R. Jin, "Semi-supervised Learning by Mixed Label Propagation" in Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 1, AAAI Press, Vancouver, British Columbia, Canada, **2007**, pp. 651–656.

[57] A. B. Goldberg, X. Zhu, S. J. Wright, "Dissimilarity in Graph-Based Semi-Supervised Classification" in Proceedings of the 11th International Conference on Artificial Intelligence and Statistics, (Eds.: M. Meila, X. Shen), Journal of Machine Learning Research - Proceedings Track, **2007**, pp. 155–162.

[58] G. Patterson, "SUN Attribute Database: Discovering, Annotating, and Recognizing Scene Attributes" in Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Washington, DC, USA, **2012**, pp. 2751–2758.

[59] Y. Liu, R. Jin, L. Yang, "Semi-supervised Multi-label Learning by Constrained Non-negative Matrix Factorization" in Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1, AAAI Press, Boston, Massachusetts, **2006**, pp. 421–426.

[60] A. Budanitsky, G. Hirst, "Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures", *Workshop on WordNet and Other Lexical Resources Second meeting of the North American Chapter of the Association for Computational Linguistics Pittsburgh USA* **2001**.

[61] A. S. Lauring, R. Andino, "Quasispecies Theory and the Behavior of RNA Viruses", *PLoS Pathog* **July 2010**, *6*, e1001005.