A Survey on Learning Approaches for Undirected Graphical Models. Application to Scene Object Recognition.

Jose-Raul Ruiz-Sarmiento[a,*], Cipriano Galindo[a], Javier Gonzalez-Jimenez[a]

[a]*Machine Perception and Intelligent Robotics Group, System Engineering and Auto. Dept., University of Málaga, Campus de Teatinos, 29071, Málaga, Spain.*

Abstract

Probabilistic Graphical Models (PGMs) in general, and Undirected Graphical Models (UGMs) in particular, become suitable frameworks to capture and conveniently model the uncertainty inherent in a variety of problems. When applied to real world applications, such as scene object recognition, they turn into a reliable and widespread resorted tool. The effectiveness of UGMs is tight to the particularities of the problem to be solved and, especially, to the chosen learning strategy. This paper presents a review of practical, widely resorted learning approaches for Conditional Random Fields (CRFs), the discriminate variant of UGMs, which is completed with a thorough comparison and experimental analysis in the field of scene object recognition. The chosen application for UGMs is of particular interest given its potential for enhancing the capabilities of cognitive agents. Two state-of-the-art datasets, NYUv2 and Cornell-RGBD, containing intensity and depth imagery from indoor scenes are used for training and testing CRFs. Results regarding success rate, computational burden, and scalability are analyzed, including the benefits of using parallelization techniques for gaining in efficiency.

Keywords: undirected graphical models, conditional random fields, parameters learning, training, scene object recognition

1. Introduction

Intelligent cognitive agents, *e.g.* mobile robots, aiming to successfully operate in human environments require the ability to *understand* what is going on in their surroundings. Scene object recognition systems are cornerstone components for such ability, providing the robot with high-level information that can be used for tasks like *scene understanding* [1, 2] or *semantic mapping* [3, 4]. The limited resources normally available in these agents force recognition systems to perform efficiently, while also dealing with the uncertainty latent in both, the robots' sensory system and their models of the working environment.

Probabilistic Graphical Models (PGMs) [5] in general, and Undirected Graphical Models (UGMs) in particular, also known as Markov Random Fields (MRFs) [6], offer suitable frameworks to tackle such uncertainty, incorporating contextual relations among the scene objects. Briefly, they rely on a graph representation to model the perceived objects as random variables in the form of nodes, and the relations among them as edges (see Fig. 1). Over this graph model, object recognition can be efficiently conducted by means of *probabilistic inference* queries. Previous to such inference, a *learning phase* must be completed in order to tune the model numerical parameters, also called weights, for the application at hand.

Typically, the learning phase in MRFs is targeted at maximizing the *expected likelihood* of the model with respect to a set of training data. However, computing this likelihood requires *exact inference*, which is in general a $\mathcal{NP}$-hard problem [5, 7]. Two major approaches stand out to overcome this concern: (i) the definition of alternative, tractable objective functions, or (ii) the estimation of the likelihood by approximate inference algorithms [8, 9, 10]. The performance of methods from both options highly differs depending on the domain of the problem at hand, *i.e.* the nature and internal structure of the data to work with. Therefore, for a given application, a thorough study is needed in order to obtain a successful model.

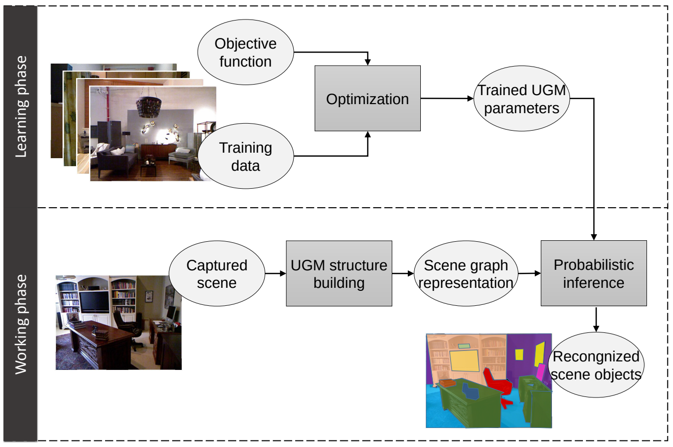

Figure 1: Learning and working phases involving Undirected Graphical Models for scene object recognition. Boxes are processes, while ovals represent consumed/produced data. This work focuses on the selection of different objective functions and optimization techniques for scene object recognition using training data from the NYUv2 and Cornell-RGBD datasets.

*Corresponding author
Email addresses: `jotaraul@uma.es` (Jose-Raul Ruiz-Sarmiento), `cipriano@ctima.uma.es` (Cipriano Galindo), `javiergonzalez@uma.es` (Javier Gonzalez-Jimenez)

In this work we present a review of the most resorted learning approaches and empirically analyze their performance in the scope of the scene object recognition. The aim of this study is to serve as a guide to quickly set-up a working-system as successful as possible for such problem. Concretely, we focus on the analysis of the following objective functions for tuning the parameters of Conditional Random Fields (CRFs) [11, 12], the discriminative variant of MRFs:

- The *pseudo-likelihood* function [13], as an alternative to the expected likelihood, and
- The most popular approximate inference algorithms for estimating the likelihood, including *Marginal* queries (Sum-product Loopy Belief Propagation [14]) and *Maximum a Posteriori* (MAP) queries (Iterated Conditional Modes [13], Graph-cuts [15], Max-product Loopy Belief Propagation [16]).

To complete the learning phase, an optimization process is needed to estimate the model parameters. Again, there is not an *ultimate* method, given that their performance depends on the chosen objective function to optimize and the problem domain, so two alternatives successfully applied in the related literature are explored in this work: the Stochastic Gradient Descent (SGD) [17] method, and the quasi-Newton Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [18] one.

In order to test the trained CRFs, thorough evaluations are carried out using the Undirected Probabilistic Graphical Models in C++ library (UPGMpp) [19] and two widely-used datasets in robotics: NYUv2 (New York University Depth Dataset version 2) [20] and Cornell-RGBD (Cornell University RGB-D Dataset) [21]. Both datasets contain labeled intensity and depth imagery from indoor scenes, albeit they show distinctive characteristics whose influence in the learning phase is studied: while NYUv2 comprises a high number of labeled images (we have used 208 from home environments) that capture the objects and relations from portions of scenes (see Fig. 2-middle), Cornell-RGBD provides a lower number of scenes (28 from homes) but fully covering the inspected place (see Fig. 2-left), which results in a considerably larger number of perceived objects and relations.

The presented study focuses on two facets of the learning methods: the recognition performance of the trained CRFs, and the required computational time. To measure the CRFs performance we have executed different MAP inference methods over the learned models, and compared their recognition results with the ground-truth information provided by the datasets. The combination of Marginal inference and SGD for learning yielded the best recognition results, achieving a success of $\sim$ 80% and $\sim$ 67% in NYUv2 and Cornell-RGBD respectively. The computational time needed by each learning method to converge is also analyzed, studying the advantage of parallelization techniques. Results of the achieved speed-up are shown employing the Open Multi-Processing API (OpenMP) [22]. Finally, the scalability of the learning methods according to different factors is also studied.

2. Related work

In the last decade, the utilization of Probabilistic Graphical Models (PGMs) [5] for tackling the scene object recognition problem has become increasingly popular. This is, to a great extent, due to their suitability to efficiently model this type of problems, while facing uncertain sensory data and contextual relations [23, 24, 25]. PGMs are divided into directed and undirected models, being the latter more natural for modeling some problems such as image analysis and spatial statistics[1] [5, 26]. As a result, numerous works have come out to explore different learning and inference methods for such models. In this section we first briefly review the most commonly used learning approaches (Sec. 2.1) and discuss some relevant works addressing scene object recognition through Unditected Graphical Models (UGMs) (Sec. 2.2), next we show examples of datasets and software tools in the field (Sec. 2.3), and finally comment previous studies on the applicability of UGMs to different problem domains (Sec. 2.4).

2.1. Learning apporaches for UGMs

In order to learn an UGM, it is typically needed an objective function containing the model parameters, and an optimization method to estimate such parameters. Regarding the objective function, and given the intractability of the expected likelihood one, there are two commonly adopted solutions: to estimate the likelihood function by approximate inference algorithms, or the utilization of alternative objective functions. Exact inference approaches like forward-backward and Viterbi algorithms [27], required for the computation of the expected likelihood, can be only applied to simple UGM structures like chains or trees. This is not the usual case when dealing with real-world scenarios, hence the utilization of approximate inference methods. There are two major queries to be solved by these methods, namely Marginal and Maximum a Posteriori (MAP) queries. Some inference techniques have variants for answering both of them, like sampling-based Markov Chain Monte Carlo (MCMC) [28] approaches (*e.g.* Gibbs sampling [29]), or variational methods like Mean Field [30] or Loopy Belief Propagation [14, 16]. Among the methods for answering MAP queries, we can find local search algorithms like Iterated Conditional Modes (ICM) [13], Graph-cuts based ones [15], or techniques from linear programming [30]. Additionally, in cases where the training data is *partially observed*, the Expectation-Maximization (EM) algorithm [31] provides the basis for the approximation of the likelihood function, although in this work we will focus on datasets containing *fully observed* data.

Concerning the alternative objective functions, perhaps the most resorted one in the literature is the pseudo-likelihood [13] due to its known virtues [32, 33, 34], although other alternatives exist. An example of this is the Piecewise likelihood [35, 36], which learns each element in the UGM graph separately, but

[1]Nevertheless, Directed Graphical Models (*e.g.* Bayesian Networks) can be converted into undirected ones maintaining the same independence assumptions if a chordal graph representation can be built [5]. In that cases, the learning approaches reviewed here are also applicable.

that is not consistent for all models, that is, its performance largely depends on the graph. Another example are Score Matching [37] and Ratio Matching [38], which although consistent, may lack the desired *convexity property* in optimization problems.

Once the objective function is defined, the most popular optimization techniques for optimizing its parameters are those working with its gradient, like the Stochastic Gradient Descent (SGD) [17] method, or the quasi-Newton Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [18] one. There are alternative techniques that do not work directly on the gradient, like those based on Generalized Iterative Scaling (GIS) [39] or Improved Iterative Scaling (IIS) [40], which have a long tradition in statistics, or those relying on Voted Perceptron [41]. However, these techniques have been outperformed by gradient-based approaches [42, 43].

Given the large number of possible combinations of objective functions and optimization methods, in this work we have selected those extensively used and providing better results in the literature, aiming to reduce this number while keeping under the spotlight the more relevant learning strategies.

2.2. UGMs for scene object recognition

First works in this regard captured the scene through RGB imagery. For instance, the work by Quattoni *et al.* [44] proposed a Conditional Random Field (CRF) for the recognition of parts of objects, performing both learning and inference through Belief Propagation. Xiang *et al.* [45] also considered a CRF trained by an ad-hoc procedure to solve a set of independent quadratic programming problems, and resorted to the ICM option for inference. Joint object recognition and scene categorization is explored by Murphy *et al.* [46] utilizing a CRF with parameters learned through a quasi-Newton method. Depth information coming from stereo has been also studied, like in the work presented by Floros and Leibe [47] that proposes a CRF ensuring the consistency among the scene object categories, and exploits the Graph-cuts technique to perform inference.

Since the arrival of Kinect-like devices, several works have come to spotlight performing a richer modeling of the scene through RGB-D imagery, being CRFs the most chosen option. Some examples are the works by Wolf *et al.* [48], where both inference and training processes of the proposed CRF are based on the Mean Field approximation, and the one by Husain *et al.* [49], where a CRF is built upon a point cloud representation of the scene and trained using LBP.

The model in Ruiz-Sarmiento *et al.* [50, 51] is learned by optimizing the pseudo-likelihood function, and applied to recognize objects through an ICM inference process. The same authors extended the CRF to also consider contextual relations between objects and rooms in [52]. Pseudo-likelihood and ICM are also resorted by Xiong *et al.* [53] to classify planar patches into coarse categories. Markov Random Fields (MRFs) have been also explored with this aim, like in Anand *et al.* [21], where a model isomorphic to a MRF is built from the segmented regions of a point cloud and training is tackled by a Graph-cuts based procedure, as in the case of Xiaofeng *et al.* [54].

2.3. RGB-D datasets and software applications

The irruption of proposals exploiting RGB-D information has been accompanied with public datasets that offer common benchmarking resources for comparing these works. Among them we can find Berkely-3D [55], Cornell-RGBD [56], NYUv1 [57], NYUv2 [20], TUW [58], SUN3D [59], or ViDRILO [60]. Specially, we highlight Cornell-RGBD, which is employed in several works aforementioned [21, 61, 49], and NYUv2 used in [48, 50, 51, 52]. The interest on this kind of data also favored the emergence of software to handle them, like SmartAnnotator [62] or the Object Labeling Toolkit (OLT) [63]. It is also worth to mention the effort done in the development of software libraries implementing efficient versions of UGM related algorithms, as in the case of the UGM Matlab toolbox [64], CRFsuite [65], or the Undirected Probabilistic Graphical Models in C++ library (UPGMpp) [19].

2.4. Analysis of learning approaches

We can find in the literature studies of the suitability of different UGM training approaches when applied to certain problems. For example, Kumar *et al.* [8] conducted an experimental comparison of several learning and inference algorithms in the context of image denoising applications on 2D image lattices. Korč and Förstner [9] analyzed the performance of CRFs trained with the pseudo-likelihood objective function to face the same problem, and compared it with a penalized variant of such objective function and the methods explored in [8]. In the work presented by Parise and Welling [10], a number of algorithms were tested over binary valued pairwise MRF models, namely pseudo-likelihood, Contrastive Divergence, Loopy Belief Propagation and Pseudo-moment Matching. Finley and Joachims [66] used six multi-label datasets from different domains, like outdoor natural scenes or text recognition, to analyze methods for training fully connected MRFs: Greedy Search, Loopy Belief Propagation, Linear-programming Relaxation, and Graph-cuts. These studies do not establish an always-winning learning strategy, since each problem domain has its own peculiarities favoring the performance of different approaches. Thus, a study of the suitability of different learning methods dealing with the scene object recognition problem comes up profitable for refining the UGMs performance. To the best of our knowledge, this study is currently missing in the literature.

This paper covers this gap by conducting an empirical analysis of the most popular training strategies when applied to the scene object recognition problem. Two families of objective functions are explored: pseudo-likelihood, and approximate inference algorithms[2], including Marginal and Maximum a Posteriori methods: sum-product and max-product Loopy Belief Propagation, Iterated Conditional Modes, and Graph-cuts. Two approaches for the optimization of such objectives are also considered: Stochastic Gradient Descent, and Limited-memory

[2]For the sake of simplicity, we will refer to the strategies that estimate the expected likelihood by means of approximate inference methods just as approximate inference approaches.

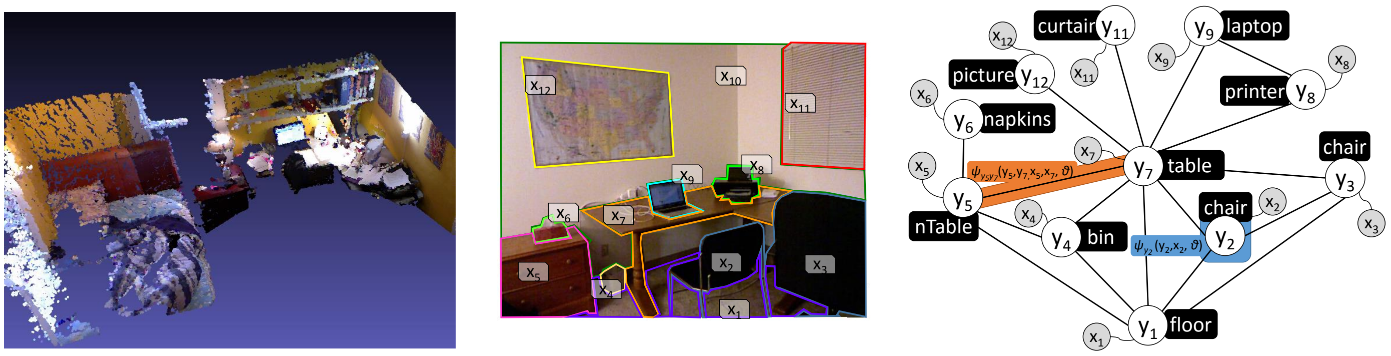

Figure 2: Left, excerpt of a home scene from the Cornell-RGBD dataset. Middle, Scene from the NYUv2 dataset with segmented patches. Right, Conditional Random Field (CRF) graph built according to the patches in the NYUv2 scene (the node and relations of the wall, x_{10}, have been omitted for clarity). The orange area illustrates the scope of a pairwise factor, while the blue one stands for the scope of an unary factor. Black boxes represent the expected results from a probabilistic inference process over such CRF.

Broyden-Fletcher-Goldfarb-Shanno. As a testbed for the conducted analysis we have employed the indoor home scenes from the NYUv2 and Cornell-RGBD, which exhibit particular features worth to explore.

3. Conditional Random Field Models for Scene Object Recognition

Conditional Random Fields (CRFs), first proposed by Lafferty *et al.* [11], are discriminative models that work with the conditional probability distribution $p(\boldsymbol{y}|\boldsymbol{x})$, in other words, they model the probability of a set of random variables $\boldsymbol{y}$ conditioned on a number of input observations $\boldsymbol{x}$. In the scope of the scene object recognition problem, the input data $\boldsymbol{x} = [x_1, ..., x_n]$ are the set of n object observations, while the random variables $\boldsymbol{y} = [y_1, ..., y_n]$ take values from a set of possible categories of that objects, denoted as $\mathcal{L}$ (*e.g.* table, vase, picture, etc.). CRFs are defined by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with a set of nodes $\mathcal{V}$ representing the random variables in $\boldsymbol{y}$, and a number of undirected edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ linking nodes that are related in some way. For example, in the addressed problem two nodes are connected if their associated objects in the scene are closer than a certain distance threshold (see the graph representation in Fig. 2-right). This means that, for a certain object, only the nearby objects in the scene have a direct influence on its recognition. According to the Hammersley-Clifford theorem [67], the probability $p(\boldsymbol{y}|\boldsymbol{x})$ can be factorized over the graph $\mathcal{G}$ as a product of *factors* $\psi(\cdot)$:

$$p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{x}, \boldsymbol{\theta})} \prod_{c \in C} \psi_c(y_c, x_c, \boldsymbol{\theta}) \tag{1}$$

where C is the set of maximal cliques[3] of the graph $\mathcal{G}$, and $Z(\cdot)$ is the also called partition function, which plays a normalization role so $\sum_{\xi(\boldsymbol{y})} p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) = 1$, being $\xi(\boldsymbol{y})$ a possible assignation to the variables in $\boldsymbol{y}$. The vector $\boldsymbol{\theta}$ stands for the model parameters (or weights) to be tuned during the learning phase. Factors can be considered as functions encoding a piece of $p(\boldsymbol{y}|\boldsymbol{x})$ over a part of the graph. Typically two kind of factors are considered: *unary factors* $\psi_i(y_i, x_i, \boldsymbol{\theta})$, which refer to nodes and talk about the probability of a random variable y_i belonging to a category in L, and *pairwise factors* $\psi_{ij}(y_i, y_j, x_i, x_j, \boldsymbol{\theta})$ that are associated with edges and state the compatibility of two random variables (y_i, y_j) being tied to a certain pair of categories. This election of factors entails the utilization of cliques with at most two nodes (see orange and blue portions in Fig. 2-right). Given that these factors must be always positive, the expression in Eq.1 can be equivalently expressed employing log-linear models and exponential families as [30]:

$$p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{x}, \boldsymbol{\theta})} \prod_{c \in C} \exp(\langle \phi(x_c, y_c), \boldsymbol{\theta} \rangle) \tag{2}$$

being $\langle \cdot, \cdot \rangle$ the inner product, and $\phi(x_c, y_c)$ the sufficient statistics of the factor over the clique c. These sufficient statistics comprise the salient features of the data, and are specific to each problem domain. In our case, examples of features characterizing nodes in $\mathcal{V}$ are the height of the object, orientation, color, etc., while features about the relations in $\mathcal{E}$ are difference in height, difference of orientation, etc.

For a better understanding of the next section, it is worth to introduce the more compact, *canonical form* of Eq.2 [68], which is obtained by grouping together all the sufficient statistics into a single vector $\phi(\boldsymbol{y}, \boldsymbol{x})$:

$$p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) = \exp\left(\langle \phi(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{\theta} \rangle - Z_l(\boldsymbol{x}, \boldsymbol{\theta})\right) \tag{3}$$

where $Z_l(\cdot)$ is the log-partition function, i.e:

$$Z_l(\boldsymbol{x}, \boldsymbol{\theta}) = \ln \sum_{\xi(\boldsymbol{y})} \exp(\langle \phi(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{\theta} \rangle) \tag{4}$$

4. Learning the model

The most common approach to tackle the learning phase of a CRF consists of finding the vector of parameters $\boldsymbol{\theta}$ that maximizes the likelihood in Eq.2, also known as Maximum Likeli-

[3]A maximal clique is a fully-connected subgraph that can not be enlarged by including an adjacent node.

hood Estimation (MLE), with respect to a certain i.i.d. training dataset $\mathcal{D} = [d_1, \dots d_m]$. That is:

$$\max_{\boldsymbol{\theta}} \mathcal{L}_p(\boldsymbol{\theta} : \mathcal{D}) = \max_{\boldsymbol{\theta}} \prod_{i=1}^{m} p(\boldsymbol{y}^i \mid \boldsymbol{x}^i; \boldsymbol{\theta}) \\ = \max_{\boldsymbol{\theta}} \exp\Big(\sum_{i=1}^{m} [\langle \phi(\boldsymbol{x}^i, \boldsymbol{y}^i), \boldsymbol{\theta} \rangle) - Z_l(\boldsymbol{x}^i; \boldsymbol{\theta})]\Big) \tag{5}$$

Each training sample $d_i = (\boldsymbol{x}^i, \boldsymbol{y}^i)$ corresponds to a scene with a number of observed objects $(\boldsymbol{x}^i)$ and ground truth information about their categories $(\boldsymbol{y}^i)$.

In order to reduce the undesirable effects of over-fitting, a prior distribution $p(\boldsymbol{\theta})$ can be defined over the model parameters. The most commonly used is a Gaussian prior with zero-mean and the same variance for all the parameters, that is $P(\theta_i \mid \sigma^2) = (1/\sqrt{2\pi\sigma})\exp(-\theta_i^2/2\sigma^2)$, so the objective function to deal with, expressed for computational convenience as the negative log-likelihood, turns to be[4]:

$$\mathcal{NLL}_p(\boldsymbol{\theta} : \mathcal{D}) = \frac{\| \boldsymbol{\theta} \|_2^2}{2\sigma^2} - \sum_{i=1}^{m} (\langle \phi(\boldsymbol{x}^i, \boldsymbol{y}^i), \boldsymbol{\theta} \rangle - Z_l(\boldsymbol{x}^i; \boldsymbol{\theta})) \tag{6}$$

Notice that maximizing $\mathcal{L}_p(\boldsymbol{\theta} : \mathcal{D})$ (once the prior $p(\boldsymbol{\theta})$ have been introduced) is equivalent to minimizing $\mathcal{NLL}_p(\boldsymbol{\theta} : \mathcal{D})$. In the former case, the objective function is ensured to be concave, while in the latter one it is convex [69]. The integration of priors of this nature is also called *L2-regularization*, since the penalization imposed to the model parameters is measured in the Euclidean or L2-norm. Intuitively, this regularization penalizes parameters taking large values, dealing thus with the over-fitting problem.

The gradient of function Eq.6 with respect to the model parameters $\boldsymbol{\theta}$ is normally needed in the optimization process, and is computed as:

$$\frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{NLL}_p(\boldsymbol{\theta} : \mathcal{D}) = \frac{\boldsymbol{\theta}}{\sigma^2} - \sum_{i=1}^{m} (\phi(\boldsymbol{x}^i, \boldsymbol{y}^i) - \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x}^i;\boldsymbol{\theta})}[\phi(\boldsymbol{x}^i, \boldsymbol{y})]) \tag{7}$$

where the expectation $\mathbb{E}_{p(\boldsymbol{y}|x^i;\boldsymbol{\theta})}[\phi(x^i, \boldsymbol{y})]$ stands for the partial derivative of the log-partition function, which is retrieved by:

$$\mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})}[\phi(\boldsymbol{x}, \boldsymbol{y})] = \sum_{\xi(\boldsymbol{y})} p(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta}) \phi(\boldsymbol{x}, \boldsymbol{y}) \tag{8}$$

In this way, the gradient of a certain parameter θ_i corresponds to the difference between the empirical expectation of its associated sufficient statistics and its expectation according to the distribution $p(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta})$, resulting in zero when they are the same.

Taking a closer look at Eq.6 and Eq.7 we can realize that they require the computation of the partition function $Z_l(\cdot)$, which in practice turns this optimization into a $\mathcal{NP}$-hard, unfeasible problem. Two major approaches arise in the literature to face this issue [10]: resort to alternative, more tractable objective functions, or to estimate the likelihood by approximate inference algorithms. Next, we briefly describe the approaches analyzed in our study: an alternative function, the pseudo-likelihood (Sec. 4.1), and two types of inference algorithms, namely Marginal inference (Sec. 4.2.1) and Maximum a Posteriori (MAP) inference (Sec. 4.2.2). The considered methods to optimize shuch objective functions conclude the section (Sec. 4.3).

4.1. Alternative objective functions: Pseudo-likelihood

The utilization of the pseudo-likelihood (PL) function, first proposed by Besag [13], was one of the earliest methods to tackle the complexity of the learning phase. It requires the computation of the likelihood of each individual random variable in $\boldsymbol{y}$ conditioned on a full observation of the rest of variables. Mathematically:

$$p(y_i \mid \boldsymbol{y}_{N_{\mathcal{G}}(y_i)}, \boldsymbol{x}; \boldsymbol{\theta}) = \frac{\psi_i(y_i, x_i, \boldsymbol{\theta})}{Z_{PL}(\boldsymbol{x}, \boldsymbol{\theta})} \prod_{y_j \in N_{\mathcal{G}}(y_i)} \psi_{ij}(y_i, y_j, x_i, x_j, \boldsymbol{\theta}) \tag{9}$$

being $N_{\mathcal{G}}(y_i)$ the neighbor of the node i in the graph $\mathcal{G}$, and the PL partition function:

$$Z_{PL}(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{\xi(y_i)} \psi_i(y_i, x_i, \boldsymbol{\theta}) \prod_{y_j \in N_{\mathcal{G}}(y_i)} \psi_{ij}(y_i, y_j, x_i, x_j, \boldsymbol{\theta}) \tag{10}$$

Thereby, $Z_{PL}(\cdot)$ becomes a *local partition function*, given that it sums over all the possible categories for the node i, unlike the original $Z(\cdot)$ that considers all the possible categories for all the nodes. Replacing the original conditional probability in Eq.5 by the one in Eq.9 results in the alternative optimization problem:

$$\max_{\boldsymbol{\theta}} \mathcal{L}_{PL}(\boldsymbol{\theta} : \mathcal{D}) = \max_{\boldsymbol{\theta}} \prod_{i=1}^{m} \prod_{j \in \mathcal{V}^i} p(y_j^i \mid \boldsymbol{y}_{N_{\mathcal{G}}(y_j^i)}, \boldsymbol{x}^i; \boldsymbol{\theta}) \tag{11}$$

An interesting feature of the PL function is that it tends to the likelihood function when the number of training samples m goes to infinity [32]. However, if a limited number of samples is considered, both objective functions might differ and their optimization provide divergent results.

In order to optimize Eq.11, it is more convenient to employ its negative log-likelihood form, along with its gradient with respect to $\boldsymbol{\theta}$ expressed by:

$$\frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{NLL}_{PL}(\boldsymbol{\theta} : \mathcal{D}) = \\ \frac{\boldsymbol{\theta}}{\sigma^2} - \sum_{i=1}^{m} \sum_{j \in \mathcal{V}^i} (\phi(\boldsymbol{x}_j^i, \boldsymbol{y}_j^i) - \mathbb{E}_{p(y_j^i|y_{N_{\mathcal{G}}(y_j^i)}, x_j^i;\boldsymbol{\theta})}[\phi(x_j^i, y_j^i)]) \tag{12}$$

where the expectation in the second term can be computed as:

$$\mathbb{E}_{p(y_i|y_{N_{\mathcal{G}}(y_i)} x_i;\boldsymbol{\theta})}[\phi(x_i, y_i)]] = \sum_{\xi(y_i)} p(y_i \mid \boldsymbol{y}_{N_{\mathcal{G}}(y_j^i)}, x_i, \boldsymbol{\theta}) \phi(x_i, y_i) \tag{13}$$

[4]Note that the constant $\ln(1/\sqrt{2\pi\sigma})$ has been omitted in the equation since it has no effect on the optimization.

It has been observed that the optimization of the pseudo-likelihood objective tends to give more importance to the parameters associated with the pairwise relations, causing the learned model to be a poor solution [12, 70]. This effect can be partially mitigated by the introduction of a regularization factor, as pointed out in the conducted analyses (see Sec. 5).

4.2. Approximate inference

An alternative approach is to replace the exact solutions of the probability queries demanded by the negative log-likelihood calculation (recall Eq.6 and Eq.7) by the outcome of an approximate inference method. In this work we consider two different types of approximations: those provided by Marginal inference methods, and the ones by MAP inference algorithms.

4.2.1. Marginal inference

Given a conditional probability distribution $p(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta})$, *Marginal inference* methods provide an approximation to the log-partition function Z_l, which is required in Eq.6, and to the marginal probabilities of its random variables in $\boldsymbol{y}$ to be plugged in Eq.7. The success of this learning approach depends largely on the *ability* of the chosen Marginal inference algorithm to converge to a valid solution.

Loopy Belief Propagation (sum-product). The sum-product version of LBP [71] is a widely-used algorithm to answer this type of probability queries. Briefly, it is based on the exchange of statistical information among the nodes in the graph according to their relations. For example, the message from node y_i to node y_j, denoted as $m_{ij}(y_j)$, indicates the beliefs of node y_i about the belonging category of node y_j. It is computed by:

$$m_{ij}^t = \psi_{ij}(y_i, y_j, x_i, x_j, \boldsymbol{\theta})\, \psi_i(y_i, x_i, \boldsymbol{\theta}) \prod_{\boldsymbol{y}_k \in N_{\mathcal{G}}(\boldsymbol{y}_i) \setminus \boldsymbol{y}_j} m_{ki}(\boldsymbol{y}_i) \tag{14}$$

where $N_{\mathcal{G}}(y_i) \setminus y_j$ is the set of neighbors of y_i in the graph $\mathcal{G}$ less y_j, and t is an iteration counter. The LBP algorithm keeps sending messages until the graph is calibrated, *i.e.* the messages of two consecutive iterations are the same. Once calibrated, the belief of each node is computed as:

$$b(y_i) = \alpha\, \psi_i(y_i, x_i, \boldsymbol{\theta}) \prod_{y_j \in N_{\mathcal{G}}(\boldsymbol{y}_i)} m_{ji} \tag{15}$$

α being a normalization component so the beliefs for node y_i sum to 1. These beliefs correspond to the marginal probabilities of the node y_i, and are employed in Eq.7 during the learning phase. Finally, the log-partition function Z_l, needed in Eq.6, is approximated by the Bethe Free energy [72] as follows:

$$Z_l(\boldsymbol{x}, \boldsymbol{\theta}) = \Big(\sum_{y_i \in \mathcal{V}} \langle b(y_i), \ln b(y_i) \rangle - \sum_{(y_i, y_j) \in \mathcal{E}} \langle b(y_i, y_j), \ln b(y_i, y_j) \rangle \Big) - \Big(\sum_{y_i \in \mathcal{V}} \langle b(y_i), \ln(\phi(y_i, x_i)\, \theta_i) \rangle - \sum_{(y_i, y_j) \in \mathcal{E}} \langle b(y_i, y_j), \ln(\phi(y_i, y_j, x_i, x_j)\, \theta_{ij}) \rangle \Big) \tag{16}$$

where $ln(\cdot)$ is the natural logarithm of each element in the vector, and $b(y_i, y_j)$ the belief of the edge linking both nodes.

An important caveat of this algorithm is that it could not converge, or does it to an approximate answer, which can produce inaccurate and oscillating gradient estimations. This instability causes final results to depend on the point at which the algorithm stops, hence compromising the convergence of the overall process. This is specially relevant in line-search methods (see Sec. 4.3.2) where the evaluations of the gradients at different points in the search can be inconsistent [5].

4.2.2. Maximum a Posteriori inference

The goal of a MAP query is to find the most probable assignment $\hat{\boldsymbol{y}}$ to the variables in $\boldsymbol{y}$, that is:

$$\hat{\boldsymbol{y}} = arg \max_{y} p(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta}) \tag{17}$$

Notice that the computation of the partition function $Z(\cdot)$ is needed but, since given a certain CRF graph its value remains constant, this expression can be simplified by:

$$\hat{\boldsymbol{y}} = arg \max_{y} \prod_{c \in C} \exp(\langle \phi(x_c, y_c), \boldsymbol{\theta} \rangle) \tag{18}$$

Nevertheless, this task is still intractable because it needs to check every possible assignment to the variables in $\boldsymbol{y}$. To tackle this problem we have resorted to three widely-used, approximate MAP inference methods, namely Iterative Conditional Modes, Graph-cuts, and the max-product version of LBP.

Once the MAP approximation is estimated by one of these methods, it can be used in Eq.7 during the learning phase. On the other hand, the log-partition function in Eq.6 is approximated by the unnormalized log-likelihood of the MAP assignment, $Z_l^{MAP} = \langle \phi(\boldsymbol{x}, \xi^{MAP}(\boldsymbol{y})), \boldsymbol{\theta} \rangle$. This approach is often called *Viterbi training* [5]. As, in general, the computation of a MAP assignment requires less computational effort than other probability queries, this practice is very appealing for a variety of real problems.

A known effect of this approach is that it produces discontinuities in the gradient estimation. A MAP assignment to a random variable y_i can be viewed as giving a probability of 1 to its most probable belonging category, so the remaining categories take a null probability. Thus, according to Eq.7, only the gradients related to such *winning* category will be updated. This fact will be studied in the evaluation section.

Iterated Conditional Modes. ICM [13] tackles the inference problem by maximizing local conditional probabilities instead of the full $p(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta})$, in a similar fashion to pseudo-likelihood. For that, it initializes the assignments to the variables in $\boldsymbol{y}$ to some initial object categories, typically those maximizing the unary factors, and iterates over such nodes (according to a scheduled order) changing the state of the variables to these, maximizing the local conditional probability:

$$\hat{y}_i = arg \max_{y_i} p(y_i \mid y_{N_{\mathcal{G}}(y_i)}, \boldsymbol{x}; \boldsymbol{\theta}) \tag{19}$$

The algorithm execution ends when convergence is accomplished, which occurs when an iteration over all the nodes is completed without changing the state of any variable, or when a predefined maximum number of iterations is reached. In practice only a few iterations are needed to converge to the MAP estimation.

Graph-Cuts (α-expansions). Techniques based on Graph-cuts [73], like α-expansions, reduce the MAP inference task to instances of the minimum cut problem. The outcome of these min-cuts are used to *expand* each of the object categories α in $\mathcal{L}$, *i.e.* to change the category assigned to a random variable from $\bar{\alpha} \in \mathcal{L}_{-\alpha}$ to α, until no expansion exists that increases the expected likelihood.

Being $\mathcal{V}_\alpha$ the set of nodes assigned to the object category α, and $\mathcal{V}_{\bar{\alpha}}$ the nodes assigned to other categories, graph cuts are computed in each iteration of the α-expansions algorithm to retrieve the minimum cut of the graph $\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c)$, where $\mathcal{V}_c = \{\mathcal{V}_{\bar{\alpha}}, s, t\}$, and $\mathcal{E}_c = \{e_{ij} \mid (y_i \neq \alpha) \cap (y_j \neq \alpha)\} \cup \{e_{sk}, e_{kt}, \forall k \in \mathcal{V}_{\bar{\alpha}}\}$. Note that for this computation two extra nodes are added to $\mathcal{V}_c$, which are usually called source (s) and sink (t). Regarding $\mathcal{E}_c$, it is compounded by the edges in $\mathcal{G}$ among the nodes that have not been assigned to the category α, and the edges connecting each of these nodes with both the source and the sink. After computing the minimum cut, the nodes still connected to the source produce an α-expansion to category α, while those linked with the sink keep their initial categories. This process is repeated until convergence is achieved, that is, no α-expansion can be done that increases the value of the expected likelihood, or until a limit number of iterations is reached. However, as in the case of ICM, this algorithm usually converges in a few iterations, and most of the expansions are performed in the early steps.

Loopy Belief Propagation (max-product). The LBP algorithm previously introduced can be also applied to MAP inference resorting to its *max-product* variant [16]. Concretely, it performs a max-product rather than sum-product message passing:

$$m_{ij}^t = \max_{y_i} \psi(y_i, y_j, x_i, x_j)\psi(y_i, x_i) \prod_{y_k \in N_{\mathcal{G}}(y_i) \setminus y_j} m_{ki}(\mathbf{y}_i) \quad (20)$$

The belief $b(y_i)$ of each individual variable y_i is computed employing the same equation as in the sum-product case (recall Eq.15). Once retrieved, the MAP assignment to y_i takes the value of the category with the highest belief.

4.3. Optimization of the objective function

So far we have described the objective functions considered in this work to estimate the model parameters $\boldsymbol{\theta}$. In this section we *complete* the learning phase (recall Fig. 1) with the discussion of some of the most used optimization methods to perform such estimation. We start by describing the Stochastic Gradient Descent algorithm in Sec. 4.3.1, along with some variants that work fairly well in practice. Sec. 4.3.2 outlines other popular optimizer for training CRFs, Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS).

Algorithm 1 Stochastic Gradient Descent

```
1:  procedure SGD(
        $\boldsymbol{\theta}^1$,        ▷ Initial assignment to the parameters in $\boldsymbol{\theta}$
        $t^{max}$,        ▷ Maximum number of iterations
        $\delta$)        ▷ Convergence threshold
2:      $t \leftarrow 1$
3:      repeat
4:          $\mathcal{D}_S \leftarrow subset(\mathcal{D})$        ▷ Get a subset from $\mathcal{D}$
5:          $g^{t+1} \leftarrow \mathbf{0}$
6:          for each $d_i \in \mathcal{D}_S$ do
7:              $g^{t+1} \leftarrow g^{t+1} + \frac{\partial}{\partial \theta^t} \mathcal{NLL}_p(\boldsymbol{\theta}^t : d_i)$
8:          $\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t - \eta g^{t+1}$        ▷ Update parameters
9:          $t \leftarrow t + 1$
10:     until ($\delta > \parallel \boldsymbol{\theta}^t - \boldsymbol{\theta}^{t-1} \parallel$) or ($t > t^{MAX}$)
11:     return ($\boldsymbol{\theta}^t$)
```

4.3.1. Stochastic Gradient Descent

SGD is a stochastic approximation of the traditional descent gradient optimization algorithm [17]. The pseudo-code for this method is depicted in Alg. 1. Briefly, it starts with an initial assignment to the model parameters. Then, a subset of samples $\mathcal{D}_S$ from the training dataset $\mathcal{D} = [d_1, \ldots d_m]$ is selected, and the parameters are updated according to a learning rate η, also called step size, and the gradients of the negative log-likelihood of these samples. The subtraction in line 8 of Alg. 1 indicates that we are moving in the opposite direction of the gradient, *i.e.* trying to reach a minimum. Typically, this process is repeated until the difference between the parameters of two consecutive iterations is lower than a given threshold, or a maximum number of iterations t^{max} is reached. This way to check convergence (line 10) is very sensitive to oscillating gradient computations, breaking the algorithm execution without reaching a valid model. A solution for this is to compare the parameters at iteration t with those computed at n iterations before t, this is $\delta > \parallel \boldsymbol{\theta}^t - \boldsymbol{\theta}^{t-n} \parallel$.

The process of selecting the subset $\mathcal{D}_S \subseteq \mathcal{D}$ (line 4) controls the randomness of the algorithm. In fact, if $\mathcal{D}_S = \mathcal{D}$, we are in the case of the standard gradient descent method. On the contrary, if $\mid \mathcal{D}_S \mid = 1$, the algorithm is said to perform in a *fully online* fashion, updating the parameters based on individual data samples. The arbitrary cardinality of the subset $\mathcal{D}_S$, and the various criteria that can be used to choose its elements, leads to a large number of different algorithm variants.

In its turn, the learning rate η has a crucial effect on the optimization performance. If η is too large, updates of the parameters $\boldsymbol{\theta}$ could be abrupt, resulting in non-convergence. Otherwise, if η is small, the convergence will be probably achieved, but the algorithm will perform slowly requiring a higher number of iterations. Therefore, a reasonable trade-off between computational time and convergence is to utilize large values of η when the algorithm starts, and progressively decrease the value as long as the iterations go on. More efficient variants of this algorithm, some of them commented below, propose different schedules to dynamically fit this learning rate.

Algorithm 2 Broyden-Fletcher-Goldfarb-Shanno

```
1:  procedure BFGS(
        $\theta^1$,             ▷ Initial assignment to the parameters in $\theta$
        $t^{max}$,              ▷ Maximum number of iterations
        $\delta$,               ▷ Convergence threshold
        $H_0^{-1}$)             ▷ Initial value of the inverse Hessian
2:      $t \leftarrow 1$
3:      repeat
4:          $g^{t+1} \leftarrow \frac{\partial}{\partial\theta^t} \mathcal{NLL}_p(\theta^t : \mathcal{D})$          ▷ Search direction
5:          $d^t(\theta^t) \leftarrow H_t^{-1} g^{t+1}$
6:          $\eta^t \leftarrow arg\min_{\eta\geq 0} \mathcal{NLL}_p(\theta^t - \eta d^t : \mathcal{D})$
7:          $\theta^{t+1} \leftarrow \theta^t - \eta d^t$          ▷ Update parameters
8:          $s^{t+1} \leftarrow \theta^{t+1} - \theta^t$          ▷ Store increments
9:          $z^{t+1} \leftarrow g^{t+1} - g^t$
10:         $H_{t+1}^{-1} \leftarrow BFGSHessianApprox(H_t^{-1}, s^{t+1}, z^{t+1})$
11:         $t \leftarrow t + 1$
12:     until $(\delta > \| \theta^t - \theta^{t-1} \|)$ or $(t > t^{MAX})$
13:     return $(\theta^t)$
```

Scheduled SGD. This is the simplest variant of SGD proposing a dynamic value for the learning rate η [74]. It is built on the idea of decreasing η according to the current algorithm iteration. This can be done in different ways, being the rule adopted in this work:

$$\eta^t \leftarrow \eta \left(1 - \ln\left[(e-1)\frac{t}{t^{max}} + 1\right]\right) \tag{21}$$

where t^{max} stands for the maximum number of iterations.

Momentum SGD. The momentum variant [75] aims to accelerate the gradient descent by trying to move in the direction of previous parameter updates, avoiding in this way possible gradient oscillations. This results in an two-steps update:

$$\begin{aligned} \Delta\theta^{t+1} &\leftarrow \eta g(\theta^t) + \alpha\Delta\theta^t \\ \theta^{t+1} &\leftarrow \theta^t - \Delta\theta^{t+1} \end{aligned} \tag{22}$$

where $\alpha \in [0, 1]$ is the momentum coefficient.

Stochastic Meta-Descent. The goal of the Stochastic Meta-Descent (SMD) algorithm [76] is to accelerate the convergence by using second-order information to adapt the learning rates. These rates are retrieved by an update with meta-gain μ:

$$\boldsymbol{\eta}_{t+1} \leftarrow \boldsymbol{\eta}_t \max(\frac{1}{2}, 1 - \mu \boldsymbol{g}_{t+1}\boldsymbol{v}_{t+1}) \tag{23}$$

being the auxiliary vector $\boldsymbol{v}$ an indicator of the dependence of the parameters on gain history, regulated as:

$$\boldsymbol{v}_{t+1} \leftarrow \lambda\boldsymbol{v}_t - \boldsymbol{\eta}_t(\boldsymbol{g}_t + \lambda \mathbf{H}_t \boldsymbol{v}_t) \tag{24}$$

where λ is another scaling parameter, and the hessian $\mathbf{H}_t$ is computed efficiently by forward-mode algorithmic differentiation [77].

4.3.2. Limited-memory Broyden-Fletcher-Goldfarb-Shanno

Second-order methods can improve convergence, but they require the computation of the Hessian matrix $\mathbf{H}(\theta) = \frac{\partial^2}{(\partial\theta)^2}\mathcal{NLL}_p(\theta^t : \mathcal{D})$, which is often unfeasible. The L-BFGS algorithm is a quasi-Newton method that constructs a sequence of matrices that approximate the Hessian and its inverse. The pseudo-code of the classical BFGS version [78] is shown in Alg. 2. We can see how, conversely to the SGD, there is no selection of a subset of $\mathcal{D}$ to compute the gradients. In contrast, it works in a *batch* mode where all the training data is employed in each iteration (line 4). Moreover, the search direction is now set by the Hessian (line 5), while the learning rate is dynamically retrieved by a line-search method (line 7) [79]. The approximation of the inverse of the Hessian is performed in line 10, where the algorithm makes use of the past approximations and the current increments of the parameters $\boldsymbol{\theta}$ and gradients $\boldsymbol{y}$ in the following way:

$$\mathbf{H}_{t+1}^{-1} = (I - \rho^t z^t s^{tT})\mathbf{H}_t^{-1}(I - \rho^t s^t z^{tT}) + \rho^t z^t s^{tT} \tag{25}$$

where $\rho^t = (z^{tT}s^t)^{-1}$. Notice that the inverse Hessian of the past iteration, $\mathbf{H}_t^{-1}$, can be recursively computed, although, in general, they are not stored in memory when a large number of parameters are involved.

The L-BFGS variant additionally improves memory efficiency and computational time by only taking into account information from the last m iterations, namely $s^t, s^{t-1}, \ldots s^{t-m-1}$ and $z^t, z^{t-1}, \ldots z^{t-m-1}$. The interested reader can refer to the work by Jorge Nocedal for further information [18].

5. Results

This section presents the findings of the conducted experimental analysis of the previously described CRF learning strategies applied to the scene object recognition problem. We first describe in Sec. 5.1 the datasets that have served as a testbed for this study, and then elucidate the different outcomes of the analysis focusing on: recognition success (Sec. 5.2), computational time (Sec. 5.3), and scalability (Sec. 5.4). Notice that there is a myriad of possible configurations of both objective functions and optimization methods, so we have discussed here the ones showing a higher impact on the learning phase performance.

5.1. Testbed

In this work we have resorted to two popular datasets providing RGB-D information: NYUv2 [20] and Cornell-RGBD [56]. The first one contains a total of 1,449 labeled pairs of both intensity and depth images, and has been extensively used in the literature (*e.g.* [48, 50, 51, 52]) due to its challenging, cluttered scenes from commercial and residential buildings. In accordance with the scope of this paper, we have employed 208 scenes captured from home facilities. A total of 24 object categories typically appearing in such environments have been considered, e.g: bottle, cabinet, counter, faucet, floor, mirror, sink, toilet, towel, table, sofa, book, etc. It is worth to mention that the provided images only capture a portion of the scene, so the

Table 1: Features characterizing the objects and contextual relations in NYUv2 and Cornell-RGBD. Values in parentheses give the number of features grouped under the same feature's name.

Object features	
NYUv2	**Cornell-RGBD**
Orientation	Orientation
Planarity	Planarity
Linearity	Linearity
Minimum height	Scatter
Centroid height	Centroid height
Volume	Horizontal extent
Area of the biggest face	Vertical extent
Hue variation	HSV means (3)
	HoG features (31)

Contextual relation features	
NYUv2	**Cornell-RGBD**
Perpendicularity	Perpendicularity (2)
Vertical distance	Vertical distance
Volume ratio	Horizontal distance
On/under relation	Coplanarity
Bias	Closest distance

contained contextual relations are somehow limited. An evidence of this is given by the total number of extracted relations, 1, 345, when compared with the number of objects, 1,295. This is an average of 6.25 objects and 6.47 relations per scene. The first column of Tab. 1 summarizes the features (sufficient statistics) that characterize the objects and contextual relations in this dataset.

The Cornell-RGBD repository has 24 labeled office scenes and 28 home labeled scenes built from the registration of RGB-D images. As opposed to NYUv2, the provided data inspect a larger portion of the scene, resulting in a richer set of available contextual information. This feature has motivated its utilization in a variety of works (e.g [21, 61, 49]). As before, we have resorted to the home scenes, which sum up a total of 764 object instances and 2,911 contextual relations among them, averaging 27.29 objects and 103.96 relations per scene. We have considered the same set of 17 object categories as used in the work presenting this dataset [21]. The second column of Tab. 1 lists the features employed in this case.

5.2. Scene object recognition results

The recognition performance of the trained CRFs has been obtained through cross-validation [80]. For each CRF working with the NYUv2 dataset, a five-fold cross-validation process was carried out, that is, the 208 scenes were randomly shuffled into 5 groups, 4 of them used for training and the remaining one for testing. Testing is performed by executing the MAP inference methods introduced in Sec. 4.2.2 (ICM, Graph-cuts and LBP) over the trained CRFs. Finally, the recognition performance is retrieved by averaging the results of 200 repetitions of such process. In the case of the Cornell-RGBD dataset, given the reduced number of scenes (28), we have resorted to a leave-one-out cross-validation process, where a scene is randomly selected for testing and the remaining 27 serve as training data. This process is also repeated 200 times. We have resorted to the implementations of the algorithms discussed in this work within the Undirected Probabilistic Graphical Models library (UPPGMpp) [19].

The recognition results are shown in Tab. 2, where the columns index different learning strategies, *i.e.* combinations of objective functions and optimization methods, while rows index CRFs configurations. The first configuration consist of CRFs only presenting nodes, and their performance set a baseline to evaluate the success of more elaborated configurations.

Notice that the results of Marginal and MAP inference methods for learning employing L-BFGS are missing in this table. This is because L-BFGS is usually unable to converge to a solution of these objective functions and, when it converges, the performance is much lower than the reported by SGD. The reason for this is the oscillating and discontinuous gradient estimations computed by these methods, as it was commented in Sec. 4.2.1 and Sec. 4.2.2. The next sections discuss: i) how to normalize the input features to increase the CRFs performance, ii) the effect of L2-regularization, iii) an analysis of Marginal inference methods for training/testing, and finally iv) some general remarks.

5.2.1. Feature engineering

It is well known that the selection of discriminant features to describe the scene objects and their relationships is crucial for learning successful models. However, despite a suitable choice of features, if their values considerably vary in magnitude, the performance of learning methods can be seriously hampered. This can be understood recalling the role of features (sufficient statistics) in Eq.7, and it is specially relevant when dealing with CRFs with edges (contextual information). In fact, as shown in the third configuration in Tab. 2, the exploitation of edge features among objects does not necessarily lead to an increase in the recognition performance. To face this, we have found extremely useful to normalize the feature values so most of them lay on the interval $[0 \cdots 10]$. For example, let's suppose a vector $\boldsymbol{f_x}$ containing the height from the floor of all the objects within the training data. To remove possible spurious from the equation, its 5th percentile (P_5) and the 95th one (P_{95}) are computed. Thus, the normalized value of this feature for each object x_i is computed as:

$$\bar{f}_{x_i} = \frac{10(f_{x_i} - P_5)}{P_{95} - P_5} \tag{26}$$

The great benefits of this normalization can be checked in the fourth configuration of Tab. 2, showing a sharp increment in the performance for whatever combination of learning strategies and inference methods. On the other hand, the second row of this table reports the results employing only nodes and normalization. Note that this increment is more pronounced when employing the NYUv2 dataset, given that in the Cornell-RGBD one several features were already normalized.

Table 2: Scene object recognition results of CRFs trained with different combinations of objective functions and optimization methods. Rows index different configurations and inference algorithms, while columns index learning strategies. Bold numbers remark the most successful inference method for a given learning strategy and configuration. N = nodes, E = edges, FN = feature normalization, R = L2-Regularization.

		NYUv2 dataset				Cornell RGBD dataset			
		L-BFGS	SGD			L-BFGS	SGD		
	Inference method	PL	PL	Marginal	MAP	PL	PL	Marginal	MAP
N	ICM, GC, LBP	**65.47%**	**38.19%**	**37.98%**	**28.54%**	**52.36%**	**54.70%**	**55.44%**	**50.18%**
N+FN	ICM, GC, LBP	**67.92%**	**66.58%**	**66.53%**	**53.44%**	**52.41%**	**53.05%**	**55.63%**	**50.97%**
N+E	ICM	**57.89%**	**22.54%**	43.24%	51.99%	**56.51%**	45.55%	**29.38%**	**11.58%**
	Graph cuts	35.72%	12.26%	39.93%	**45.36%**	48.70%	**47.75%**	**29.38%**	6.67%
	LBP	40.30%	17.24%	**51.05%**	44.43%	41.17%	46.71%	**29.38%**	5.89%
N+E+FN	ICM	69.11%	67.17%	73.42%	68.66%	56.14%	55.39%	58.33%	58.46%
	Graph cuts	69.12%	68.52%	71.64%	67.24%	55.18%	55.43%	58.94%	56.83%
	LBP	**73.14%**	**75.53%**	**79.85%**	**74.43%**	**59.94%**	**59.88%**	**66.37%**	**60.92%**
N+E+FN+R	ICM	76.47%	72.5%	74.74%	73.13%	62.32%	61.76%	64.24%	62.95%
	Graph cuts	75.66%	71.80%	73.23%	70.79%	60.93%	58.83%	65.05%	59.50%
	LBP	**78.80%**	**76.08%**	**79.69%**	**74.35%**	**65.02%**	**63.02%**	**67.27%**	**65.13%**

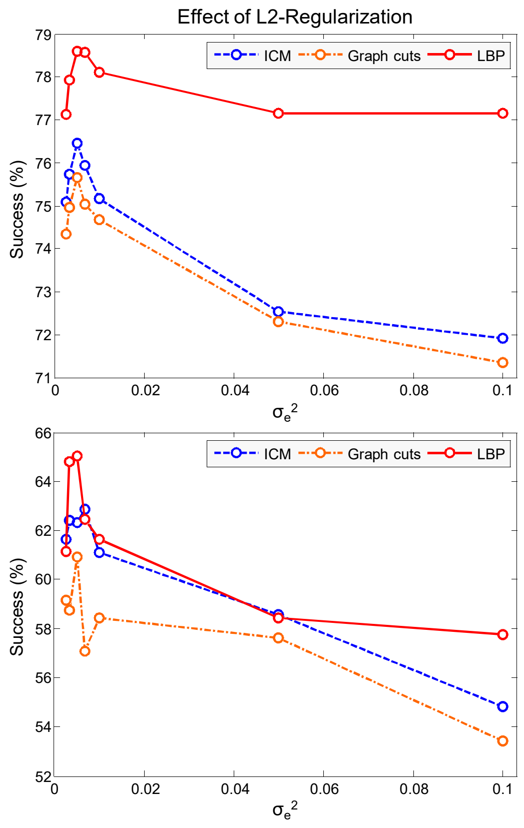

Figure 3: Effect of L2-regularization on the performance of CRFs trained with a pseudo-likelihood – L-BFGS strategy in NYUv2 (top) and Cornell-RGBD (bottom) datasets. The standard deviation of parameters associated with node features is fixed to 0.1.

5.2.2. *The effect of regularization*

As commented in Sec. 4, a way to avoid the over-fitting of the CRF parameters is to use regularization. To illustrate its effect on the recognition performance, we have trained a number of CRFs introducing a L2-regularization term (recall Eq.6), whose outcomes are depicted in the last configuration of Tab. 2. These results reveal that regularization and learning approaches resorting to pseudo-likelihood are good partners, especially in the case of L-BFGS optimization, where it boosts the performance more than a 5% in both datasets. On the other hand, the strategies based on Marginal/MAP inference and SGD also benefit from regularization in the Cornell-RGBD dataset, while in NYUv2 the impact is reduced. This indicates that CRFs working with Cornell-RGBD become more complex models than those dealing with NYUv2, so it is more probable to over fit their parameters. At this point it is also worth to recall one of the core differences between NYUv2 and Cornell-RGBD: the number of contextual relations among objects. In NYUv2 this number is reduced in comparison with Cornell-RGBD, however, we can see how that information is enough to properly exploit the objects' context.

In order to provide these results, a study for each learning strategy was carried out to properly tune the standard deviation value in the regularization term. This analysis arose that approaches employing pseudo-likelihood yield better results when two standard deviations are considered, one for the parameters associated with node features (σ_n^2), and other one for those related to edge features (σ_e^2), being $\sigma_n^2 >> \sigma_e^2$. This matches the behaviour described in Sec. 4.1: the pseudo-likelihood optimization tends to give more importance to the parameters associated with edge features, so it makes sense to penalize more those parameters. For example, Fig. 3 illustrates the performance reached by the three MAP inference methods

Table 3: Recognition results for CRFs (with edges and normalized features) trained with MAP inference plus SGD. Rows index the MAP inference methods used during training, while columns index the testing methods. Bold numbers highlight the best results for each learning strategy.

	NYUv2 dataset			Cornell RGBD dataset		
	ICM	Graph-cuts	LBP	ICM	Graph-cuts	LBP
ICM	73.35%	69.85%	**74.45**%	**58.9**%	49.99%	46.32%
Graph-cuts	70.80%	**73.32**%	55.31%	39.80%	**53.25**%	16.12%
LBP	68.66%	67.24%	**74.43**%	58.46%	56.83%	**60.92**%

over CRFs trained by a pseudo-likelihood – L-BGGS strategy, being $\sigma_n^2 = 0.1$, and σ_e^2 ranging from 0.0025 up to 0.1. In both datasets the best results come with $\sigma_e^2 = 0.005$.

Combinations resorting to Marginal or MAP inference get better results when σ_n^2 and σ_e^2 are similar, so in these cases the same σ can be used to *regularize* all the parameters. Concretely the results reported in Tab. 2 were obtained with $\sigma = 0.125$ for strategies with Marginal inference, and $\sigma = 0.2$ for those with MAP inference.

5.2.3. Marginal inference methods for learning

As introduced in Sec. 4.2.2, the core of learning approaches resorting to MAP inference is the chosen method to perform such probability query, being LBP the one employed to report their results in Tab. 2. In the literature it is suggested that the best performance of these approaches is achieved when the same method used during training is also used for testing [5, 8]. We have conducted the analysis shown in Tab. 3 aiming to check if this also holds in the scene object recognition problem. The expected result is to find the most proficient CRF models on the diagonals of such table, which is perfectly fulfilled in the Cornell-RGBD case. Regarding the NYUv2 dataset, the combination ICM-LBP slightly outperforms the ICM-ICM one by $\sim 1\%$, but since there are no discrepancies in the other combinations, we can expect a reliable performance when resorting to the same MAP inference algorithm for learning and testing.

5.2.4. General remarks

It is worth to mention the accomplishment of the LBP inference method for retrieving the object recognition results from tuned CRFs. This is the winning option when the features are normalized or regularization is performed (see Fig. 3 and the fourth and fifth configurations in Tab. 2), showing in most cases a considerable improvement with respect to ICM and Graph-cuts. It is also noteworthy the results reported by the Marginal inference – SGD combination, which is the winning strategy in both datasets for the last two configurations. Moreover, every learning strategy has at least one configuration where it is unable to tune valid models, with the exception of the PL – LBFG-S one, which is more robust in that sense.

The reader may have noticed that the recognition performance of CRFs working with different datasets have not been compared. This is because neither the same object categories nor objects/relations features are shared between the two datasets, so a fair comparison in this regard is not possible.

Table 4: Average time spent by the winning learning strategies in Tab. 2 for tuning the model parameters.

	L-BFGS	SGD		
Dataset	PL	PL	Marginal	MAP
NYUv2	36.21s.	28.32s.	69.17s.	40.11s.
Cornell-RGBD	31.13s.	20.55s.	72.89s.	72.24s.

5.3. Computational time

Apart from the recognition success, another important factor to evaluate the suitability of a learning strategy is the required time to tune a model. This is especially relevant for cognitive agents operating in human environments, like mobile robots, where the learning phase could be performed several times out of the laboratory. Such a re-learning is necessary, for example, to include object categories not appearing in the dataset, objects showing peculiar configurations, etc., that is, situations that are detected during the agent operation in a particular workspace [81]. If the learning phase takes too long, the agent can be operating with an obsolete representation of such knowledge, so fast learning strategies are preferred.

Tab. 4 reports the computational time needed for the winning learning strategies from Tab. 2. These figures were yielded by a computer with an Intel Core i7-3820 at 3.60GHz. microprocessor and a memory of 2x4GB. DDR3 at 1,600MHz. running the optimized implementation of the algorithms within the UPGMpp library.

We can see how, in general, computational times are reduced, being the fastest strategy pseudo-likelihood – SGD. This combination spends on average 28.32s. to tune a CRF model with the NYUv2 dataset, and 20.55s with the Cornell-RGBD one. The Marginal inference – SGD combination, which achieved the highest recognition success in both datasets, is in this case the slowest one, taking 69.17s. and 72.89s. for NYUv2 and Cornell-RGBD respectively. Despite these times are still short, depending on the application it could be needed a trade-off between time and recognition success motivating the choice of a faster strategy.

In the remaining of this section we: i) survey the application of more sophisticated SGD algorithms to accelerate convergence, ii) study how to speed-up even more the learning process by means of parallelization techniques, and iii) discuss the effect of the peculiarities of NYUv2 and Cornell-RGBD on the learning computational time.

Table 5: Computational time required by different SGD variants to train a CRF model. It is also shown the speed-up achieved with respect to the standard SGD, and the maximum value of the initial learning rate that tunes valid models.

	NYUv2				Cornell-RGBD			
	Standard	Schedule	Momentum	Meta-descent	Standard	Schedule	Momentum	Meta-descent
Computational time	69.17s.	48.32s.	55.25s.	43.50s.	72.89s.	60.08s.	65.63s.	52.89s.
Speed-up factor	-	x1.43	x1.25	x1.59	-	x1.21	x1.11	x1.38
Initial learning rate (η)	10^{-5}	30^{-5}	10^{-5}	50^{-5}	10^{-5}	30^{-5}	20^{-5}	75^{-5}

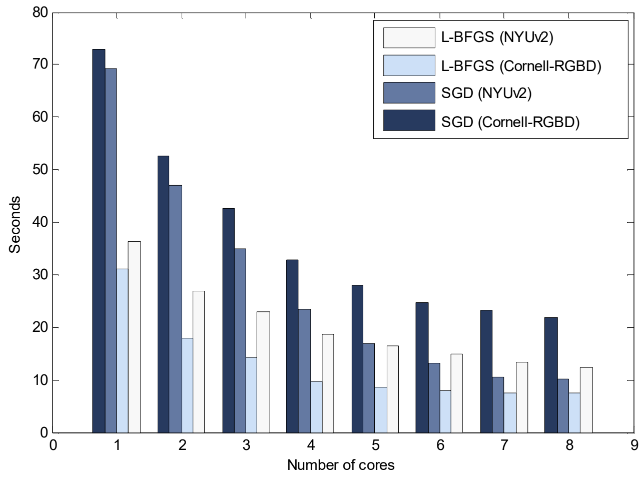

Figure 4: Time spent by the standard and parallelized versions of two learning combinations: pseudo-likelihood – L-BFGS, and Marginal – SGD.

5.3.1. SGD methods

As commented in Sec. 4.3.1, the standard SGD algorithm employs a fixed learning rate η while optimizing the model parameters, being $\eta = 10^{-5}$ in the experiments reported so far. More sophisticated SGD variants dynamically adjust such learning rate in order to accelerate convergence, as in the case of Schedule, Momentum and Meta-descent SGD. Tab. 5 compares the training times reported by those methods. In both datasets the algorithm reaching the highest speed-up is Meta-descent, followed by Schedule, and Momentum in the last position but, as expected, all of them improving the results reported by the standard SGD algorithm. Tab. 5 also shows the maximum initial learning rate that makes SGD converge to valid CRF models. To complete the configuration of the studied methods, the Momentum variant achieved the best results with the momentum coefficient $\alpha = 0.4$, and the Meta-descent one with the meta-gain $\mu = 0.01$ and scaling factor $\lambda = 0.5$.

5.3.2. Parallelization

In this section we explore the utilization of parallelization techniques in order to reduce the learning computational time required by L-BFGS and SGD based optimizations. Recalling the SGD algorithm in Alg. 1, in lines 4-7, a subset of the training dataset is selected, and the gradients of the negative log likelihood of such subset are computed. The parallelized version designed here divides that computation into threads, each one executed in parallel in a different CPU-core. Until now, all the conducted experiments resorted to a subset of size one, *i.e.* only a (randomly selected) sample of the dataset is processed in each iteration (lines 3-10). In this section we analyze the speed-up achieved by choosing subsets of different sizes and dividing the workload among the same number of cores. In its turn, the algorithm of the BFGS optimization method depicted in Alg. 2 computes the gradients of the entire dataset in line 4. In this case, the parallelized variant splits this computation into multiple cores.

Fig. 4 shows the learning computational time required by these parallelized versions for two learning strategies: pseudo-likelihood – L-BFGS, and Marginal inference – SGD. Notice that the reached speed-up is non-linear since the parallelized parts of the algorithms do not cover their entire execution. Despite of this, they reach an acceleration factor of x3.53 and x5.06 (using 8 cores) for the PL – L-BFGS and Marginal inference – SGD combinations respectively.

5.3.3. Datasets' peculiarities and computational time

As introduced in Sec. 5.1, each dataset contains a different number of training samples, object categories, and features. This causes, for example, that the number of parameters to be learned differs: 3,072 parameters in the case of the NYUv2 dataset (192 associated with node features and 2,880 with edge features), and 2,431 in the Cornell-RGBD one (697 associated with node features and 1,734 with edge features). Interestingly, these differences compensate each other, resulting in similar training times whatever dataset is used. CRFs trained with NYUv2 need on average more iterations to converge, since its training samples contain less information than those in Cornell-RGBD, but for the same reason the iterations are completed faster.

5.4. Scalability

Fig. 5 depicts the results of the conducted study on scalability, where we have focused on the two most proficient learning strategies. Results concerning the scalability with respect to the number of samples used during the CRFs learning are shown in Fig. 5(a) and Fig. 5(b). Regarding the reported computational times, we can see how the PL – L-BFGS strategy scales better (linearly in some cases) than the Marginal – SGD one, specially when the number of samples increases considerably. This is due to the fact that the operation of the latter strategy,

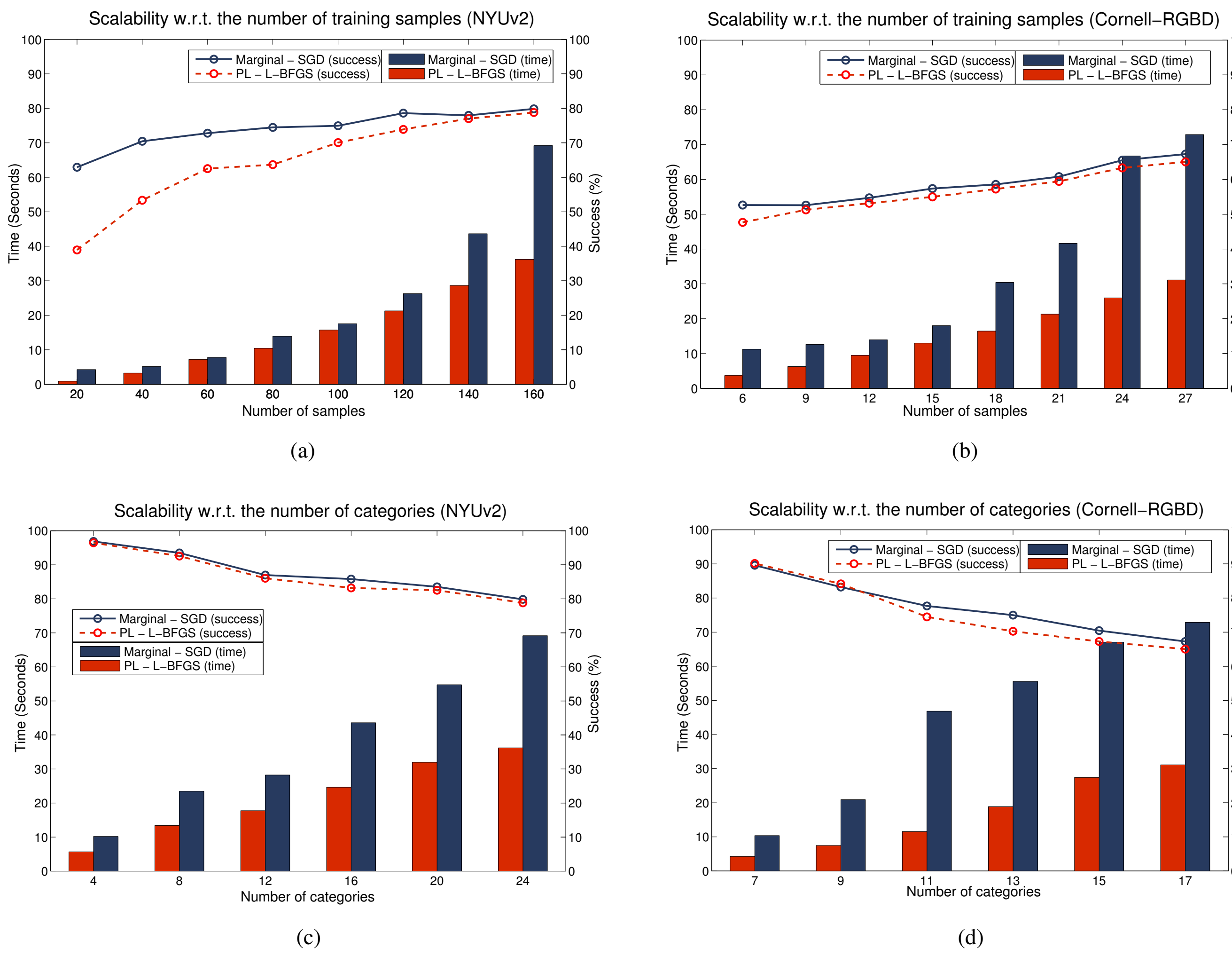

Figure 5: Analysis of how the performance of the two winning strategies, pseudo-likelihood – L-BFGS, and Marginal – SGD, scales with respect to the utilization of a different number of training samples, (a) and (b), as well as a different number of object categories, (c) and (d).

using only a training sample per learning iteration, requires progressively more time for the samples *to agree* on the model parameters when its number increases, while in the case of the PL – L-BFGS one this is partially mitigated by its batch processing (recall the algorithms in Alg. 1 and Alg. 2).

Studying the recognition success, both strategies clearly improve with the progressive addition of new training samples. In fact, the CRFs learned employing the maximum number of available samples could still improve their performance with extra data, specially in the case of the Cornell-RGBD dataset. The Marginal – SGD strategy achieves a better recognition success regardless of the number of training samples (which is more noticeable in the NYUv2 case), hence raising the idea that it requires less samples to reach the same performance as the PL – L-BFGS one. Notice that, as mentioned in the previous section, using the same number of training samples, the time needed for learning CRF models using the Cornell-RGBD dataset is higher than for that resorting to the NYUv2 one. This is due to the uneven complexity of their samples, for example, 10 training samples from Cornell-RGB have on average the same number of objects (273) as in ∼44 samples from NYUv2, while the number of contextual relations (1040) is the same as in ∼161 samples from that dataset.

Similar conclusions can be drawn from the scalability analysis with respect to the number of object categories, which results are shown in Fig. 5(c) and Fig. 5(d). Again, the computational time needed by PL – L-BFGS grows slowly (also sub-linearly in some cases) in comparison with the growth of the Marginal – SGD option. In this case, the reason for the behaviour of the latter strategy is the increasing number of model parameters, also increasing the required operations per iteration (*e.g.* gradient updates), which has a lower impact in the L-BFGS batch stance. Notice that the number of parameters does not increase linearly with the number of categories, but it follows the formula $|\boldsymbol{\theta}| = |\boldsymbol{f_n}|\times|\mathcal{L}|+|\boldsymbol{f_e}|\times|\mathcal{L}|^2$, where the operator $|\cdot|$ is overloaded to denote the size of the vectors of parameters $\boldsymbol{\theta}$, node features $\boldsymbol{f_n}$ and edge features $\boldsymbol{f_e}$, as well as the cardinality of the set of considered object categories $\mathcal{L}$. Concerning the recognition performance, it slowly decreases for both strategies when the number of object categories increases, which is the expected result for object recognition systems jointly considering multiple categories. This decrease is more pronounced in the case of the Cornell-RGBD dataset, which could be due to the utilization of less discriminative features or object categories whose instances look similar.

6. Conclusions

In this paper we have reviewed the most popular learning strategies for Conditional Random Fields (CRF) applied to the robot scene object recognition problem, conducting an in-deep experimental analysis of their performance. This particular Probabilistic Graphical Model permits us to conveniently handle the uncertainty inherent to the robot sensory system and its working space. The approaches surveyed in this paper to face the unfeasible computation of the likelihood function are divided into two groups: the definition of alternative, tractable functions (pseudo-likelihood (PL)), and the likelihood estimation through approximate (Marginal and MAP) inference methods. These objective functions have been optimized by two widely-used algorithms, namely Stochastic Gradient Descent (SGD) and Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS). The state-of-the-art datasets NYUv2 and Cornell-RGBD, both containing intensity and depth imagery from indoor domestic scenes, have been utilized as testbeds.

Results regarding recognition success were performed through cross-validation, that is, selecting a portion of those datasets for training, and another one for testing. The testing phase has been carried out by three probabilistic inference methods: Iterated Conditional Modes (ICM), Graph-cuts, and Loopy Belief Propagation (LBP). The reported analysis yields numerous findings that are summarized here:

- Approximate inference methods for learning caused L-BFGS to produce a poor model, or even to not converge, due to their oscillating and discontinuous gradient estimations.
- All the learning strategies largely benefited from the normalization of the features' values within a certain range ($[0 \cdots 10]$ in our experiments).
- CRF models learned from Cornell-RGBD data were more prone to over-fit their parameters than those working with NYUv2. This is due to the higher complexity of the scenes from the Cornell-RGBD.
- When resorting to L2-regularization to avoid such over-fitting, PL based strategies benefit from penalizing more the parameters associated with contextual features than those related to object features. This is due to the fact that PL gives more importance to contextual relations.
- The Marginal inference – SGD strategy yielded the highest recognition performance in both datasets: 79.85% in NYUv2 and 67.27% in Cornell-RGBD.
- The PL – L-BFGS strategy was the most robust, providing acceptable results in all the CRF configurations studied.
- LBP was the winning method for testing, reaching the best results when dealing with CRFs with edges and normalized features.

Another important keypoint to determinate the suitability of a learning strategy is the required computational time, which is specially relevant when it must be performed by a cognitive agent, *e.g.* a mobile robot. From our study we can extract that:

- In general, computational times are reduced, ranging from the 24.43s. (on average) of the PL – SGD strategy, up to the 71.03s. of the Marginal inference – SGD one.
- The Meta-descent variant of SGD largely accelerates the learning phase, speeding it up a factor of ~ 1.5 for the Marginal inference – SGD combination.
- L-BFGS and SGD benefited from parallelization techniques in OpenMP, achieving a speed-up factor of ~ 3.5 for PL – L-BFGS, and ~ 5 for Marginal inference - SGD.

Concerning the scalability of the studied strategies, it has been analyzed how the utilization of different number of training samples and object categories affect their performance. These experiments reported that the computational time required for learning scaled considerably better in both cases when PL – L-BFGS was used, being its growth sub-linear in some cases, while regarding recognition success, the Marginal inference – SGD option achieved the best outcome.

In the future, we plan to parallelize the considered approximate inference methods at a lower level, aiming to also take advantage of GPU cores through, for example, CUDA, accelerating even more the learning/testing phases.

Acknowledgements

This work is supported by the research projects TEP2012-530] and DPI2014-55826-R, funded by the Andalusia Regional Government and the Spanish Government, respectively, both financed by European Regional Development's funds (FEDER).

References

[1] S. Gupta, P. Arbeláez, R. Girshick, J. Malik, Indoor scene understanding with rgb-d images: Bottom-up segmentation, object detection and semantic segmentation, International Journal of Computer Vision 112 (2) (2014) 133–149.

[2] J. Yao, S. Fidler, R. Urtasun, Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, 2012, pp. 702–709.

[3] A. Pronobis, P. Jensfelt, K. Sjöö, H. Zender, G.-J. M. Kruijff, O. M. Mozos, W. Burgard, Semantic modelling of space, in: H. I. Christensen, G.-J. M. Kruijff, J. L. Wyatt (Eds.), Cognitive Systems, Vol. 8 of Cognitive Systems Monographs, Springer Berlin Heidelberg, 2010, pp. 165–221.

[4] I. Kostavelis, A. Gasteratos, Semantic mapping for mobile robotics tasks: A survey, Robotics and Autonomous Systems 66 (2015) 86–103.

[5] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, MIT Press, 2009.

[6] R. Kindermann, J. L. Snell, et al., Markov random fields and their applications, Vol. 1, American Mathematical Society Providence, RI, 1980.

[7] B. M. Marlin, N. de Freitas, Asymptotic efficiency of deterministic estimators for discrete energy-based models: Ratio matching and pseudolikelihood, in: Uncertainty in Artificial Intelligence (UAI), AUAI Press, 2011, pp. 497–505.

[8] S. Kumar, J. August, M. Hebert, Exploiting inference for approximate parameter learning in discriminative fields: An empirical study, in: Proceedings of the 5th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition, EMMCVPR'05, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 153–168.

[9] F. Korč, W. Förstner, Approximate parameter learning in conditional random fields: An empirical investigation, in: Proceedings of the 30th DAGM Symposium on Pattern Recognition, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 11–20.

[10] W. M. Parise, S., Learning in markov random fields: An empirical study, in: Proceedings of the Joint Statistical Meeting, JSM2005, 2005.

[11] J. D. Lafferty, A. McCallum, F. C. N. Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence data, in: Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001, pp. 282–289.

[12] S. Kumar, M. Hebert, Discriminative random fields, Int. J. Comput. Vision 68 (2) (2006) 179–201.

[13] J. Besag, On the statistical analysis of dirty pictures, Journal of the Royal Statistical Society. Series B (Methodological) 48 (3) (1986) 259–302.

[14] J. S. Yedidia, W. T. Freeman, Y. Weiss, Generalized Belief Propagation, in: Advances Neural Information Processing Systems, Vol. 13, 2001, pp. 689–695.

[15] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, Pattern Analysis and Machine Intelligence, IEEE Transactions on 23 (11) (2001) 1222–1239.

[16] Y. Weiss, W. T. Freeman, On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs, IEEE Trans. Inf. Theor. 47 (2) (2006) 736–744.

[17] Y. Nesterov, Introductory lectures on convex optimization : a basic course, Applied optimization, Springer US.

[18] J. Nocedal, Updating quasi-newton matrices with limited storage, in: Mathematics of Computation, Vol. 35, 1980, pp. 2376–2383.

[19] J. Ruiz-Sarmiento, C. Galindo, J. González-Jiménez, UPGMpp: a Software Library for Contextual Object Recognition, in: 3rd. Workshop on Recognition and Action for Scene Understanding, 2015.

[20] N. Silberman, D. Hoiem, P. Kohli, R. Fergus, Indoor Segmentation and Support Inference from RGBD Images, in: Proc. of the 12th European Conference on Computer Vision (ECCV 2012), 2012, pp. 746–760.

[21] A. Anand, H. S. Koppula, T. Joachims, A. Saxena, Contextually guided semantic labeling and search for three-dimensional point clouds, In the International Journal of Robotics Research 32 (1) (2013) 19–34.

[22] OpenMP Architecture Review Board: OpenMP API Specification for Parallel Programming, `http://openmp.org/wp/`, [Online; accessed 14-April-2016].

[23] C. Galleguillos, S. Belongie, Context based object categorization: A critical survey, Computer Vision and Image Understanding 114 (6) (2010) 712–722. doi:10.1016/j.cviu.2010.02.004.

[24] A. Oliva, A. Torralba, The role of context in object recognition, Trends in Cognitive Sciences 11 (12) (2007) 520–527.

[25] S. Divvala, D. Hoiem, J. Hays, A. Efros, M. Hebert, An empirical study of context in object detection, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, 2009, pp. 1271–1278.

[26] K. P. Murphy, Machine learning : a probabilistic perspective, Adaptive computation and machine learning series, MIT Press, Cambridge (Mass.), 2012.

[27] L. R. Rabiner, Readings in speech recognition, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990, Ch. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pp. 267–296.

[28] C. P. Robert, G. Casella, Monte Carlo Statistical Methods (Springer Texts in Statistics), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[29] S. Geman, D. Geman, Stochastic relaxation, gibbs distributions, and the bayesian restoration of images, IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-6 (6) (1984) 721–741.

[30] M. J. Wainwright, M. I. Jordan, Graphical models, exponential families, and variational inference, Found. Trends Mach. Learn. 1 (1-2) (2008) 1–305.

[31] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the em algorithm, JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B 39 (1) (1977) 1–38.

[32] G. Winkler, Image Analysis, Random Fields and Dynamic Monte Carlo Methods: A Mathematical Introduction, 1st Edition, Springer Publishing Company, Incorporated, 1995.

[33] P. Liang, M. I. Jordan, An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators, in: Proceedings of the 25th International Conference on Machine Learning, ICML '08, ACM, New York, NY, USA, 2008, pp. 584–591.

[34] J. K. Bradley, Learning large-scale conditional random fields, in: Dissertations, Paper 221, 2013.

[35] C. A. Sutton, A. Mccallum, Piecewise Training for Undirected Models, in: Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI-05), 2005, pp. 568–575.

[36] C. Sutton, A. McCallum, Piecewise pseudolikelihood for efficient training of conditional random fields, in: Proceedings of the 24th international conference on Machine learning, ACM, 2007, pp. 863–870.

[37] A. Hyvärinen, Estimation of non-normalized statistical models by score matching, The Journal of Machine Learning Research 6 (2005) 695–709.

[38] A. Hyvärinen, Some extensions of score matching, Journal Computational Statistics and Data Analysis 51 (5) (2007) 2499–2512.

[39] J. N. Darroch, D. Ratcliff, Generalized iterative scaling for log-linear models, Ann. Math. Statist. 43 (5) (1972) 1470–1480.

[40] S. D. Pietra, V. D. Pietra, J. Lafferty, Inducing features of random fields, IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (4) (1997) 380–393.

[41] M. Collins, Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms, in: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10, EMNLP '02, Association for Computational Linguistics, Stroudsburg, PA, USA, 2002, pp. 1–8.

[42] R. Malouf, A comparison of algorithms for maximum entropy parameter estimation, in: Proceedings of the 6th Conference on Natural Language Learning - Volume 20, COLING-02, Association for Computational Linguistics, Stroudsburg, PA, USA, 2002, pp. 1–7.

[43] F. Sha, F. Pereira, Shallow parsing with conditional random fields, in: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1, NAACL '03, Association for Computational Linguistics, Stroudsburg, PA, USA, 2003, pp. 134–141.

[44] A. Quattoni, M. Collins, T. Darrell, Conditional random fields for object recognition, in: Advances in Neural Information Processing Systems, MIT Press, 2004, pp. 1097–1104.

[45] Y. Xiang, X. Zhou, Z. Liu, T.-S. Chua, C.-W. Ngo, Semantic context modeling with maximal margin conditional random fields for automatic image annotation, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, 2010, pp. 3368–3375.

[46] K. P. Murphy, A. Torralba, W. T. Freeman, Using the forest to see the trees: A graphical model relating features, objects, and scenes, in: S. Thrun, L. K. Saul, B. Schölkopf (Eds.), Advances in Neural Information Processing Systems 16, MIT Press, 2004, pp. 1499–1506.

[47] G. Floros, B. Leibe, Joint 2d-3d temporally consistent semantic segmentation of street scenes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2012), 2012, pp. 2823–2830.

[48] D. Wolf, J. Prankl, M. Vincze, Fast semantic segmentation of 3d point clouds using a dense crf with learned parameters, in: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 2015.

[49] F. Husain, L. Dellen, C. Torras, Recognizing point clouds using conditional random fields, in: Pattern Recognition (ICPR), 2014 22nd International Conference on, 2014, pp. 4257–4262.

[50] J. R. Ruiz-Sarmiento, C. Galindo, J. González-Jiménez, Scene object recognition for mobile robots through semantic knowledge and probabilistic graphical models, Expert Systems with Applications 42 (22) (2015) 8805–8816.

[51] J. R. Ruiz-Sarmiento, C. Galindo, J. González-Jiménez, Exploiting semantic knowledge for robot object recognition, Knowledge-Based Systems 86 (2015) 131–142.

[52] J. R. Ruiz-Sarmiento, C. Galindo, J. González-Jiménez, Joint categorization of objects and rooms for mobile robots, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015.

[53] X. Xiong, D. Huber, Using context to create semantic 3d models of indoor environments, in: In Proceedings of the British Machine Vision Confer-

ence (BMVC 2010), 2010, pp. 45.1–11.

[54] X. Ren, L. Bo, D. Fox, Rgb-(d) scene labeling: Features and algorithms, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2012), 2012, pp. 2759–2766.

[55] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, T. Darrell, A category-level 3-d object dataset: Putting the kinect to work, in: 1st Workshop on Consumer Depth Cameras for Computer Vision (ICCV workshop), 2011.

[56] A. Anand, H. S. Koppula, T. Joachims, A. Saxena, Contextually guided semantic labeling and search for three-dimensional point clouds, In The International Journal of Robotics Research 32 (1) (2013) 19–34.

[57] N. Silberman, R. Fergus, Indoor scene segmentation using a structured light sensor, in: Proceedings of the International Conf. on Computer Vision - Workshop on 3D Representation and Recognition, 2011.

[58] A. Aldoma, T. Faulhammer, M. Vincze, Automation of "ground truth" annotation for multi-view rgb-d object instance recognition datasets, in: Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, 2014, pp. 5016–5023.

[59] J. Xiao, A. Owens, A. Torralba, Sun3d: A database of big spaces reconstructed using sfm and object labels, in: Computer Vision (ICCV), 2013 IEEE International Conference on, 2013, pp. 1625–1632.

[60] J. Martinez-Gomez, M. Cazorla, I. Garcia-Varea, V. Morell, Vidrilo: The visual and depth robot indoor localization with objects information dataset, International Journal of Robotics Research.

[61] O. Kahler, I. Reid, Efficient 3d scene labeling using fields of trees, in: IEEE International Conference on Computer Vision (ICCV 2013), 2013, pp. 3064–3071.

[62] Y.-S. Wong, H.-K. Chu, N. J. Mitra, Smartannotator an interactive tool for annotating indoor rgbd images, Computer Graphics Forum 34 (2) (2015) 447–457.

[63] J. R. Ruiz-Sarmiento, C. Galindo, J. González-Jiménez, OLT: A Toolkit for Object Labeling Applied to Robotic RGB-D Datasets, in: European Conference on Mobile Robots, 2015.

[64] M. Schmidt, UGM: Matlab Code for Undirected Graphical Models, `http://www.cs.ubc.ca/ schmidtm/Software/UGM.html`, [Online; accessed 10-May-2016] (2015).

[65] N. Okazaki, Crfsuite: a fast implementation of conditional random fields (crfs), http://www.chokkan.org/software/crfsuite/, [Online; accessed 28-April-2015].

[66] T. Finley, T. Joachims, Training structural svms when exact inference is intractable, in: Proceedings of the 25th International Conference on Machine Learning, ICML '08, ACM, New York, NY, USA, 2008, pp. 304–311.

[67] J. Hammersley, P. Clifford, Markov fields on finite graphs and lattices, unpublished manuscript (1971).

[68] J. Jancsary, Approximate discriminative training of graphical models, Ph.D. thesis, Institute of Telecommunications, Vienna University of Technology (2012).

[69] O. Barndorff-Nielsen, Information and exponential families in statistical theory, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, Ltd., 1978.

[70] A. Blake, C. Rother, M. Brown, P. Perez, P. Torr, Interactive image segmentation using an adaptive gmmrf model, in: Computer Vision - ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part I, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 428–441.

[71] K. P. Murphy, Y. Weiss, M. I. Jordan, Loopy belief propagation for approximate inference: An empirical study, in: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, UAI'99, 1999, pp. 467–475.

[72] Y. Weiss, Comparing the mean field method and belief propagation for approximate inference in mrfs, Advanced Mean Field MethodsTheory and Practice.

[73] B. P. D.M. Greig, A. Seheult, Exact maximum a posteriori estimation for binary images, Journal of the Royal Statistical Society. Series B (Methodological) 51 (1989) 271–279.

[74] C. Darken, J. Moody, Fast adaptive k-means clustering: some empirical results, in: Neural Networks, 1990., 1990 IJCNN International Joint Conference on, 1990, pp. 233–238 vol.2.

[75] I. Sutskever, J. Martens, G. Dahl, G. Hinton, On the importance of initialization and momentum in deep learning, in: Proceedings of the 30th international conference on machine learning (ICML-13), 1990, pp. 1139–1147.

[76] N. N. Schraudolph, Fast curvature matrix-vector products for second-order gradient descent, Neural Computation 14 (2002) 2002.

[77] S. V. N. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, K. P. Murphy, Accelerated training of conditional random fields with stochastic gradient methods, in: In International Conference on Machine Learning, 2006, pp. 969–976.

[78] C. G. Broyden, The convergence of a class of double-rank minimization algorithms, Journal of the Institute of Mathematics and Its Applications 1 (6) (1970) 76–90.

[79] J. Nocedal, S. J. Wright, Numerical Optimization, 2nd Edition, Springer series in operations research and financial engineering, Springer, New York, 2006.

[80] S. Arlot, A. Celisse, A survey of cross-validation procedures for model selection, Statistics Surveys 4 (2010) 40–79.

[81] J. R. Ruiz-Sarmiento, C. Galindo, J. González-Jiménez, Probability and common-sense: Tandem towards robust robotic object recognition in ambient assisted living, Submitted.