Efficient Training for Pairwise or Higher Order CRFs via Dual Decomposition

Nikos Komodakis
University of Crete
http://www.csd.uoc.gr/~komod

Abstract

We present a very general algorithmic framework for structured prediction learning that is able to efficiently handle both pairwise and higher-order discrete MRFs/CRFs[1]*. It relies on a dual decomposition approach that has been recently proposed for MRF optimization. By properly combining this approach with a max-margin method, our framework manages to reduce the training of a complex high-order MRF to the parallel training of a series of simple slave MRFs that are much easier to handle. This leads to an extremely efficient and general learning scheme. Furthermore, the proposed framework can yield learning algorithms of increasing accuracy since it naturally allows a hierarchy of convex relaxations to be used for MRF inference within a max-margin learning approach. It also offers extreme flexibility and can be easily adapted to take advantage of any special structure of a given class of MRFs. Experimental results demonstrate the great effectiveness of our method.*

1. Introduction

Due to the wide applicability of discrete Markov Random Fields (MRFs), related MAP estimation algorithms have attracted a significant amount of interest in computer vision research. However, besides MAP estimation, the task of learning the parameters of a MRF plays an equally important role for successfully applying MRFs to computer vision problems (a MAP-MRF solution is often of little value if the used MRF does not properly represent the problem at hand). Due to the fact that MRF variables interact with each other, MRF learning is a difficult task (it is a characteristic example of a so-called structured prediction problem). This difficulty becomes even greater due to the computational challenges that are often raised by computer vision applications with regard to learning. For instance, many of the MRFs used in vision are of large scale. Also, the complexity and diversity of vision tasks often require the training of MRFs with complex potential functions. On top of that, during the last years the use of high order MRFs is becoming increasingly popular in vision since such MRFs are often found to considerably improve the quality of estimated solutions. Most of the MRF learning methods proposed so far in the vision literature compromise with regard to at least some of the above issues. For instance, most of these methods impose restrictions on the type of the MRF potential functions that can be used during learning, and/or can handle only pairwise MRFs [3, 9, 14, 15, 11, 10, 2].

The goal of this work is to propose a very general learning framework that addresses all of the above mentioned challenges and is applicable to a very broad class of problems. To achieve this goal the proposed framework makes use of some recent advances made on the MRF optimization side [8, 7], which it combines with a max-margin approach for learning [16]. More specifically, it makes use of a dual decomposition approach [8] that has been previously used for MAP estimation. Thanks to this approach, it essentially manages to reduce the task of training an arbitrarily complex MRF to that of training in parallel a series of simpler slave MRFs that are much easier to handle within a max-margin framework. The concurrent training of the slave MRFs takes place in a principled way through an efficient projected subgradient algorithm. This lead to a powerful learning framework that makes the following contributions compared to prior art: (1) it is able to efficiently handle not just pairwise but also high-order MRFs, (2) it does not impose any restrictions on the type of MRF potential functions that can handle or on the topology of the MRF graph, (3) the reduction to the parallel training of a series of slaves MRFs in combination with the projected subgradient method leads to a highly efficient learning scheme that is also scalable even to very large problems, (4) it allows the use of a hierarchy of convex relaxations for approximating MAP-MRF estimation within learning for structured prediction (where this hierarchy includes some widely used LP relaxations for MRF inference), thus leading to structured prediction learning algorithms of increasing accuracy, (5) it is extremely flexible and extendable since the only thing that requires from a user is to be able to compute an optimizer for a slave MRF, while everything else is taken care by the framework. It can thus be easily adapted to take advantage of the special structure that may exist in any given class of MRFs that one wishes to train.

Previous approaches such as the cutting plane method [4] require solving a very expensive LP relaxation for the full MRF per iteration. They also require a QP with a growing number of constraints to be optimized at each step

[1] In this paper, the terms Markov Random Field (MRF) and Conditional Random Field (CRF) will be used interchangeably.

(although the number of constraints is polynomially upper-bounded, it can grow large in practice). Both issues impose great computational cost especially for problems of large scale or high order that are often typical in computer vision. As a result, algorithms such as [4] are orders of magnitude slower than our method. Additionally, our method is much more general compared to approaches such as [5, 10]. The fact that it proposes the core idea that the training of a complex MRF can be decomposed in a principled manner to the parallel training of a series of slave subproblems gives to our algorithm extreme generality and flexibility. It thus handles in a unified, elegant and modular manner high-order models, models that employ tighter relaxations for improved accuracy, as well as models with any type of special characteristics (*e.g.*, submodularity). More generally, simply by properly choosing what slaves to use, one can directly apply our algorithm to an even wider range of problems, which opens new possibilities. Furthermore, some additional advantages of our method are that it is inherently *parallelizable*, it naturally handles not just a squared l_2-norm but also a very broad class of other regularizers (including sparsity inducing norms - *e.g.* l_1 - that are often crucial for learning), it offers guaranteed convergence rates in all of these cases, and it also allows for general loss functions as well as for an infeasible training set.

In the remainder of the paper we review the dual decomposition method for MAP estimation in §2 and the max-margin structured prediction approach in §3, we describe and analyze our framework in §4-§7, we present experimental results in §8, and we finally conclude in §9.

2. MRF Optimization via Dual Decomposition

Let $G = (\mathcal{V}, \mathcal{C})$ be a hypergraph consisting of a set of nodes $\mathcal{V}$ and a set of hyperedges $\mathcal{C}$. Let also $\mathbf{u} = \{u_p(\cdot)\}_{p \in \mathcal{V}}$ and $\mathbf{h} = \{h_c(\cdot)\}_{c \in \mathcal{C}}$ be two sets of functions defined respectively on the nodes and hyperedges of G. The energy of a MRF with unary potentials $\mathbf{u}$ and higher-order potentials $\mathbf{h}$ is then defined as

$$\mathrm{MRF}_G(\mathbf{x}; \mathbf{u}, \mathbf{h}) := \sum_{p \in \mathcal{V}} u_p(x_p) + \sum_{c \in \mathcal{C}} h_c(\mathbf{x}_c) \ , \qquad (1)$$

where $\mathbf{x}_c$ is used to denote the set of labels $\{x_p | p \in c\}$. In MRF optimization the goal is to find the minimum of the above energy function, which is denoted by $\mathrm{MRF}_G(\mathbf{u}, \mathbf{h})$:

$$\mathrm{MRF}_G(\mathbf{u}, \mathbf{h}) = \min_{\mathbf{x}} \mathrm{MRF}_G(\mathbf{x}; \mathbf{u}, \mathbf{h}) \qquad (2)$$

The above problem is in general NP-hard. One common way to compute approximately optimal solutions to it is by making use of convex relaxations. The dual decomposition framework in [8] provides a very general and flexible method for deriving and solving tight dual relaxations in this case. According to this framework, a set $\{G_i\}$ of sub-hypergraphs of the original hypergraph $G = (\mathcal{V}, \mathcal{C})$ is chosen such that $G_i = (\mathcal{V}_i, \mathcal{C}_i)$ and $\mathcal{V} = \cup \mathcal{V}_i, \mathcal{C} = \cup \mathcal{C}_i$. The original hard problem $\mathrm{MRF}_G(\mathbf{u}, \mathbf{h})$ (also called the *master*) is then decomposed into a set of easier to solve subproblems $\{\mathrm{MRF}_{G_i}(\boldsymbol{\theta}^i, \mathbf{h})\}$ (called the *slaves*), which are defined on these sub-hypergraphs $\{G_i\}$. Each slave MRF inherits the higher-order potentials $\mathbf{h}$ of the master MRF, but has its own unary potentials $\boldsymbol{\theta}^i = \{\theta^i_p(\cdot)\}$. The key property that these unary potentials $\boldsymbol{\theta}^i$ have to satisfy is

$$\sum_{i \in \mathcal{I}_p} \theta^i_p(\cdot) = u_p(\cdot) \ , \ \forall p \in \mathcal{V} \qquad (3)$$

where $\mathcal{I}_p$ denotes the set of indices of all sub-hypergraphs containing node p, *i.e.*,

$$\mathcal{I}_p = \{i | p \in \mathcal{V}_i\} \ . \qquad (4)$$

The above property simply expresses the fact that the sum of the unary potentials of the slaves should give back the unary potentials of the master MRF. As a result of this property the sum of the minimum energies of the slaves can be shown to always provide a lower bound to the minimum energy of the master MRF, *i.e.*, it holds $\sum_i \mathrm{MRF}(\boldsymbol{\theta}^i, \mathbf{h}) \leq \mathrm{MRF}(\mathbf{u}, \mathbf{h})$. Maximizing this lower bound by adjusting the unary potentials $\boldsymbol{\theta}^i$ (which are the dual variables in this case) gives rise to a dual relaxation for problem (2):

$$\mathrm{DUAL}_{\{G_i\}}(\mathbf{u}, \mathbf{h}) = \max_{\{\boldsymbol{\theta}^i\}} \sum\nolimits_i \mathrm{MRF}_{G_i}(\boldsymbol{\theta}^i, \mathbf{h}) \qquad (5)$$

$$\text{s.t.} \sum_{i \in \mathcal{I}_p} \theta^i_p(\cdot) = u_p(\cdot) \ . \qquad (6)$$

In this manner, simply by choosing different decompositions $\{G_i\}$ of the hypergraph G, one can derive different convex relaxations, which in practice turn out to provide very good approximations for problem (2). These include the standard marginal polytope LP relaxation for pairwise MRFs, as well as other relaxations that are much tighter.

Before proceeding we should note that the dual variables $\{\boldsymbol{\theta}^i\}$ in the above problem can be equivalently expressed in terms of another set of auxiliary variables $\{\boldsymbol{\lambda}^i\}$ as

$$\theta^i_p(\cdot) = \lambda^i_p(\cdot) + \frac{u_p(\cdot)}{|\mathcal{I}_p|} \ . \qquad (7)$$

In this case it is trivial to verify that constraint (6) for variables $\{\boldsymbol{\theta}^i\}$ translates into the following constraint for variables $\{\boldsymbol{\lambda}^i\}$

$$\sum_{i \in \mathcal{I}_p} \lambda^i_p(\cdot) = 0 \ . \qquad (8)$$

Hereafter whenever we refer to variables $\{\boldsymbol{\theta}^i\}$ we will assume that there also exist variables $\{\boldsymbol{\lambda}^i\}$ satisfying eq. (8) (and vice versa).

3. Max-margin Markov Networks

Let us now come to the issue of MRF training. To this end, let $\{\bar{\mathbf{z}}^k, \bar{\mathbf{x}}^k\}_{k=1}^K$ be a training set of K samples, where

$\bar{\mathbf{z}}^k$, $\bar{\mathbf{x}}^k$ represent respectively the input data and the label assignments of the k-th sample. We assume that the MRF instance associated with the k-th sample is defined on a hypergraph $G^k = (\mathcal{V}^k, \mathcal{C}^k)$ and both the unary potentials $\mathbf{u}^k$ and the higher-order potentials $\mathbf{h}^k$ of that MRF can be expressed in terms of feature vectors extracted from the data

$$u_p^k(x_p) = \mathbf{w}^T g_p(x_p, \bar{\mathbf{z}}^k), \;\; h_c^k(\mathbf{x}_c) = \mathbf{w}^T g_c(\mathbf{x}_c, \bar{\mathbf{z}}^k) \;\; , \quad (9)$$

where $\mathbf{w}$ is a unknown vector of parameters we seek to estimate, while $g_p(\cdot,\cdot)$ and $g_c(\cdot,\cdot)$ represent known vector-valued feature functions.

Let $\Delta(\mathbf{x}, \mathbf{x}')$ represents a dissimilarity measure between any two solutions $\mathbf{x}$ and $\mathbf{x}'$ (obviously it will hold $\Delta(\mathbf{x}, \mathbf{x}) = 0$). In a maximum margin Markov network [16] we seek a vector of parameters $\mathbf{w}$ such that the MRF energy of the desired solution $\bar{\mathbf{x}}^k$ is smaller by $\Delta(\mathbf{x}, \bar{\mathbf{x}}^k)$ than the MRF energy of any other solution $\mathbf{x}$, *i.e*.

$$\mathrm{MRF}_{G^k}(\bar{\mathbf{x}}^k; \mathbf{u}^k, \mathbf{h}^k) \leq \mathrm{MRF}_{G^k}(\mathbf{x}; \mathbf{u}^k, \mathbf{h}^k) - \Delta(\mathbf{x}, \bar{\mathbf{x}}^k) + \xi_k \;\; . \quad (10)$$

To ensure that a feasible solution $\mathbf{w}$ always exists, we have also introduced into the above constraints a slack variable ξ_k, which should ideally be equal to zero. In general, however, it will hold $\xi_k > 0$ and so we must adjust $\mathbf{w}$ such that ξ_k takes a minimal value. As a result, we end up minimizing a regularized hinge loss function of the following form

$$\min_{\mathbf{w}} \mu R(\mathbf{w}) + \sum_{k=1}^{K} \xi_k \;\; .$$

In the above formula the term $R(\mathbf{w})$ represents a regularization term, which can be chosen in many different ways. For instance, $R(\mathbf{w})$ can be set equal to $\frac{1}{2}||\mathbf{w}||^2$, or to a sparsity inducing norm such as $||\mathbf{w}||_1$. Furthermore, due to constraints (10), slack variable ξ_k equals

$$\begin{aligned} \xi_k = \; & \mathrm{MRF}_{G^k}(\bar{\mathbf{x}}^k; \mathbf{u}^k, \mathbf{h}^k) - \\ & \min_{\mathbf{x}} \left(\mathrm{MRF}_{G^k}(\mathbf{x}; \mathbf{u}^k, \mathbf{h}^k) - \Delta(\mathbf{x}, \bar{\mathbf{x}}^k) \right) . \end{aligned} \quad (11)$$

We may assume that the dissimilarity measure $\Delta(\mathbf{x}, \bar{\mathbf{x}}^k)$ decomposes in the same way as the MRF energy, *i.e*., it holds

$$\Delta(\mathbf{x}, \bar{\mathbf{x}}^k) = \sum_{p \in \mathcal{V}^k} \delta_p(x_p, \bar{x}_p^k) + \sum_{c \in \mathcal{C}^k} \delta_c(\mathbf{x}_c, \bar{\mathbf{x}}_c^k) \;\; . \quad (12)$$

Therefore, by defining the new MRF potentials $\bar{\mathbf{u}}^k$, $\bar{\mathbf{h}}^k$

$$\bar{u}_p^k(\cdot) = u_p^k(\cdot) - \delta_p(\cdot, \bar{x}_p^k) \quad (13)$$

$$\bar{h}_c^k(\cdot) = h_c^k(\cdot) - \delta_c(\cdot, \bar{\mathbf{x}}_c^k) \;\; , \quad (14)$$

the slack variable ξ_k can be reexpressed as the following loss function $L_{G^k}(\bar{\mathbf{x}}^k, \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k; \mathbf{w})$

$$\begin{aligned} & L_{G^k}(\bar{\mathbf{x}}^k, \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k; \mathbf{w}) \equiv \\ & \equiv \mathrm{MRF}_{G^k}(\bar{\mathbf{x}}^k; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) - \min_{\mathbf{x}} \mathrm{MRF}_{G^k}(\mathbf{x}; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) \\ & = \mathrm{MRF}_{G^k}(\bar{\mathbf{x}}^k; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) - \mathrm{MRF}_{G^k}(\bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) \;\; , \end{aligned} \quad (15)$$

and our objective function thus becomes equal to

$$\min_{\mathbf{w}} \mu R(\mathbf{w}) + \sum_{k=1}^{K} L_{G^k}(\bar{\mathbf{x}}^k, \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k; \mathbf{w}) \;\; . \quad (16)$$

Equality (15) simply expresses the fact that the used loss $L_{G^k}(\bar{\mathbf{x}}^k, \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k; \mathbf{w})$ equals zero only if the MRF with potentials $\bar{\mathbf{u}}^k$, $\bar{\mathbf{h}}^k$ attains its minimum energy at the desired solution $\bar{\mathbf{x}}^k$.

4. Learning via Dual Decomposition

Unfortunately, even evaluating the loss function $L_{G^k}(\bar{\mathbf{x}}^k, \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k; \mathbf{w})$ is in general intractable, let alone minimizing it. The reason for this is because it is in general NP-hard to obtain the minimum energy $\mathrm{MRF}_{G^k}(\bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k)$ used in the definition of $L_{G^k}(\bar{\mathbf{x}}^k, \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k; \mathbf{w})$. Here we propose to approximate this minimum energy with the optimum of a convex relaxation of the form (5) that has been derived based on the dual decomposition framework.

To that end, we assume that each hypergraph $G^k = (\mathcal{V}^k, \mathcal{C}^k)$ has been decomposed into sub-hypergraphs $\{G_i^k = (\mathcal{V}_i^k, \mathcal{C}_i^k)\}$ and for each sub-hypergraph G_i^k a slave MRF with unary potentials $\boldsymbol{\theta}^{(i,k)}$ and higher-order potentials $\bar{\mathbf{h}}^k$ has been defined on it. In that case the resulting convex relaxation $\mathrm{DUAL}_{\{G_i^k\}}(\bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k)$ associated with the k-th MRF problem $\mathrm{MRF}_{G^k}(\bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k)$ will be

$$\mathrm{DUAL}_{\{G_i^k\}}(\bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) = \max_{\{\boldsymbol{\theta}^{(i,k)}\}} \sum\nolimits_i \mathrm{MRF}_{G_i^k}(\boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k) \quad (17)$$

$$\text{s.t.} \sum_{i \in \mathcal{I}_p^k} \theta_p^{(i,k)}(\cdot) = \bar{u}_p^k(\cdot) \, , \quad (18)$$

where $\mathcal{I}_p^k = \{i | p \in \mathcal{V}_i^k\}$. If we now replace in (15) the optimum energy $\mathrm{MRF}_{G^k}(\bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k)$ with the optimum of the convex relaxation $\mathrm{DUAL}_{\{G_i^k\}}(\bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k)$ then it will hold that

$$\begin{aligned} & L_{G^k}(\bar{\mathbf{x}}^k, \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k; \mathbf{w}) \approx \\ & \approx \mathrm{MRF}_{G^k}(\bar{\mathbf{x}}^k; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) - \mathrm{DUAL}_{\{G_i^k\}}(\bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) \\ & = \mathrm{MRF}_{G^k}(\bar{\mathbf{x}}^k; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) - \max_{\{\boldsymbol{\theta}^{(i,k)}\}} \sum\nolimits_i \mathrm{MRF}_{G_i^k}(\boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k) \\ & = \min_{\{\boldsymbol{\theta}^{(i,k)}\}} \Big(\mathrm{MRF}_{G^k}(\bar{\mathbf{x}}^k; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) - \sum\nolimits_i \mathrm{MRF}_{G_i^k}(\boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k) \Big), \end{aligned} \quad (19)$$

where in the last equality we have used the identity $-\max_i(a_i) = \min(-a_i)$. Due to the fact that variables $\boldsymbol{\theta}^{(i,k)}$ have to satisfy constraint (18) the following equality stands in this case

$$\mathrm{MRF}_{G^k}(\bar{\mathbf{x}}^k; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) = \sum\nolimits_i \mathrm{MRF}_{G_i^k}(\bar{\mathbf{x}}^k; \boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k) \, . \quad (20)$$

By substituting this equality into (19), we get

$$L_{G^k}(\bar{\mathbf{x}}^k, \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k; \mathbf{w}) \approx$$
$$\approx \min_{\{\boldsymbol{\theta}^{(i,k)}\}} \sum_i \left(\mathrm{MRF}_{G_i^k}(\bar{\mathbf{x}}^k; \boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k) - \mathrm{MRF}_{G_i^k}(\boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k) \right)$$
$$= \min_{\{\boldsymbol{\theta}^{(i,k)}\}} \sum_i L_{G_i^k}(\bar{\mathbf{x}}^k, \boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k; \mathbf{w}) \quad (21)$$

Therefore, the final function we now need to minimize is the following one, resulting from substituting (21) into (16)

$$\min_{\mathbf{w}, \{\boldsymbol{\theta}^{(i,k)}\}} \mu R(\mathbf{w}) + \sum_k \sum_i L_{G_i^k}(\bar{\mathbf{x}}^k, \boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k; \mathbf{w}) \quad (22)$$
$$\text{s.t.} \quad \sum\nolimits_{i \in \mathcal{I}_p^k} \theta_p^{(i,k)}(\cdot) = \bar{u}_p^k(\cdot) \ . \quad (23)$$

As can be seen, the initial objective function (16) (which was intractable due to containing the hinge losses $L_{G^k}(\cdot)$) has now been decomposed into the hinge losses $L_{G_i^k}(\cdot)$ that are a lot easier to handle.

To minimize the resulting convex function we will use a *projected subgradient* algorithm. For deriving the subgradient updates, in this case it is easier to temporarily replace variables $\{\boldsymbol{\theta}^{(i,k)}\}$ with the variables $\{\boldsymbol{\lambda}^{(i,k)}\}$ as defined in eq. (7). As can be seen from eq. (8) the latter variables must belong to the following set

$$\boldsymbol{\Lambda} = \left\{ \{\boldsymbol{\lambda}^{(i,k)}\} | \sum\nolimits_{i \in \mathcal{I}_p^k} \lambda_p^{(i,k)}(\cdot) = 0 \right\} \ . \quad (24)$$

According to the projected subgradient method, variables $\mathbf{w}$, $\{\boldsymbol{\lambda}^{(i,k)}\}$ must be updated at each iteration using the following scheme

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha_t \cdot d\mathbf{w} \quad (25)$$
$$\boldsymbol{\lambda}^{(i,k)} \leftarrow \mathrm{Proj}_{\boldsymbol{\Lambda}}(\boldsymbol{\lambda}^{(i,k)} - \alpha_t \cdot d\boldsymbol{\lambda}^{(i,k)}) \ , \quad (26)$$

where $d\mathbf{w}$, $\{d\boldsymbol{\lambda}^{(i,k)}\}$ denote the components of a subgradient of the objective function (22), $\mathrm{Proj}_{\boldsymbol{\Lambda}}(\cdot)$ denotes projection onto the set $\boldsymbol{\Lambda}$, and α_t is given, *e.g.*, by

$$\alpha_t = \frac{\gamma_t}{||\{d\mathbf{w}, \ d\boldsymbol{\lambda}^{(i,k)}\}||}$$

with γ_t being a positive multiplier used at the t-th iteration satisfying $\lim_{t \to \infty} \gamma_t = 0$, $\sum_{t=0}^{\infty} \gamma_t = \infty$. Optionally, one may also choose to apply different learning rates with respect to $d\mathbf{w}$ and $\{d\boldsymbol{\lambda}^{(i,k)}\}$.

To compute a subgradient of the objective function (22), we must first compute a subgradient for $-\mathrm{MRF}_{G_i^k}(\boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k) = -\min_{\mathbf{x}} \mathrm{MRF}_{G_i^k}(\mathbf{x}; \boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k)$, which is the only non-differentiable term included in the definition of $L_{G_i^k}(\cdot)$. To that end, we will use the following well known lemma:

Lemma. *Let $f(\cdot) = \max_{m=1,\ldots,M} f_m(\cdot)$, with $f_m(\cdot)$ convex and differentiable. A subgradient of f at $\mathbf{y}$ is given by $\nabla f_{\hat{m}}(\mathbf{y})$, where $\hat{m}$ is any index for which $f(\mathbf{y}) = f_{\hat{m}}(\mathbf{y})$.*

From the above lemma it follows that a subgradient of $-\mathrm{MRF}_{G_i^k}(\boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k)$ is given by the vector $-\nabla \mathrm{MRF}_{G_i^k}(\hat{\mathbf{x}}^k; \boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k)$, where $\hat{\mathbf{x}}^{(i,k)}$ denotes a minimizer of the slave MRF for G_i^k. That vector has the following components $\hat{d}\mathbf{w}^k$, $\hat{d}\boldsymbol{\lambda}^{(i,k)}$:

$$\hat{d}\mathbf{w}^k = -\frac{\partial}{\partial \mathbf{w}} \left(\sum_p \theta_p^{(i,k)}(\hat{x}_p^{(i,k)}) + \sum_c h^k(\hat{\mathbf{x}}_c^{(i,k)}) \right)$$
$$\overset{(7),(9)}{=} -\sum_p \frac{g_p(\hat{x}_p^{(i,k)}, \bar{\mathbf{z}}^k)}{\mathcal{I}_p^k} - \sum_c g_c(\hat{\mathbf{x}}_c^{(i,k)}, \bar{\mathbf{z}}^k) \quad (27)$$

$$\hat{d}\lambda_p^{(i,k)}(\cdot) = -\frac{\partial}{\partial \lambda_p^{(i,k)}} \left(\theta_p^{(i,k)}(\hat{x}_p^{(i,k)}) \right) \overset{(7)}{=} -\left[\hat{x}_p^{(i,k)} = \cdot \right] \quad (28)$$

where $[\,\cdot\,]$ equals 1 if the expression in square brackets is satisfied, and 0 otherwise. Based on the above, the components $d\mathbf{w}$, $\{d\boldsymbol{\lambda}^{(i,k)}\}$ of the total subgradient of the objective function (22) are given by

$$d\mathbf{w} = \nabla R(\mathbf{w}) + \sum_{k,i,p} \frac{g_p(\bar{x}_p^k, \bar{\mathbf{z}}^k) - g_p(\hat{x}_p^{(i,k)}, \bar{\mathbf{z}}^k)}{\mathcal{I}_p^k}$$
$$+ \sum_{k,i,c} \left(g_c(\bar{\mathbf{x}}_c^k, \bar{\mathbf{z}}^k) - g_c(\hat{\mathbf{x}}_c^{(i,k)}, \bar{\mathbf{z}}^k) \right) \quad (29)$$

$$d\lambda_p^{(i,k)}(\cdot) = \left[\bar{x}_p^k = \cdot \right] - \left[\hat{x}_p^{(i,k)} = \cdot \right] \quad (30)$$

After the update $\boldsymbol{\lambda}^{(i,k)} \leftarrow \boldsymbol{\lambda}^{(i,k)} - \alpha_t \cdot d\boldsymbol{\lambda}^{(i,k)}$, (26) also requires the resulting variables to be projected onto the feasible set $\boldsymbol{\Lambda}$. This projection is equivalent to subtracting the average vector $\left(\sum_{i \in \mathcal{I}_p^k} \lambda_p^{(i,k)}(\cdot) \right) / \mathcal{I}_p^k$ from each vector $\lambda_p^{(i,k)}(\cdot)$ such that the sum $\sum_{i \in \mathcal{I}_p^k} \lambda_p^{(i,k)}(\cdot)$ remains equal to zero as required by the definition of $\boldsymbol{\Lambda}$. Based on this observation and the definition of $d\boldsymbol{\lambda}^{(i,k)}$ given in eq. (30), the combined update (26) reduces to

$$\lambda_p^{(i,k)}(\cdot) \ += \alpha_t \left(\left[\hat{x}_p^{(i,k)} = \cdot \right] - \frac{\sum_{j \in \mathcal{I}_p^k} \left[\hat{x}_p^{(j,k)} = \cdot \right]}{\mathcal{I}_p^k} \right). \quad (31)$$

The pseudocode of the resulting learning algorithm is shown in fig. 1.

5. Algorithmic Analysis

By applying the dual decomposition method we managed to replace each term $L_{G^k}(\cdot)$, which is the hinge loss of a complicated high-order MRF on G^k, by the terms $\{L_{G_i^k}(\cdot)\}$ that are the hinge losses of a series of simpler slave MRFs on sub-hypergraphs $\{G_i^k\}$. In this manner we have essentially achieved the following: the difficult task of

Input:
Training samples $\{\bar{\mathbf{z}}^k, \bar{\mathbf{x}}^k\}_{k=1}^K$, hypergraphs $\{G^k{=}(\mathcal{V}^k, \mathcal{C}^k)\}_{k=1}^K$
Unary and high-order feature functions $\{g_p(\cdot,\cdot)\}$, $\{g_c(\cdot,\cdot)\}$

Learning algorithm:
$\forall k,$ choose decomposition $\{G_i^k{=}(\mathcal{V}_i^k, \mathcal{C}_i^k)\}$ of hypergraph G^k
$\forall k, i,$ set $\boldsymbol{\lambda}^{(i,k)} = \mathbf{0},$ initialize $\boldsymbol{\theta}^{(i,k)}$ according to (7)
repeat
*// **optimize slave MRFs***
$\forall k, i,$ compute minimizer $\hat{\mathbf{x}}^{(i,k)}$ of slave $\mathrm{MRF}_{G_i^k}(\boldsymbol{\theta}^{(i,k)}, \hat{\mathbf{h}}^k)$
*// **update** $\mathbf{w}$*
Compute $d\mathbf{w}$ based on eq. (29)
$\mathbf{w} \leftarrow \mathbf{w} - \alpha_t \cdot d\mathbf{w}$
*// **update** $\boldsymbol{\theta}^{(i,k)}$*
$\forall k, i, p,\ \lambda_p^{(i,k)}(\cdot)\ {+}{=}\ \alpha_t{\cdot}\left(\left[\hat{x}_p^{(i,k)}{=}\cdot\right] - \frac{\sum_{j\in\mathcal{I}_p^k}\left[\hat{x}_p^{(j,k)}{=}\cdot\right]}{\mathcal{I}_p^k}\right)$
Update $\boldsymbol{\theta}^{(i,k)}$ using (7)
until convergence

Fig. 1: Pseudocode of the learning algorithm.

training an arbitrarily complex high-order MRF has been reduced to the much easier task of training *in parallel* a series of simpler slave MRFs.

At a high level the algorithm operates as follows to achieve this: it allows each slave MRF to have its own unary potentials $\boldsymbol{\theta}^{(i,k)}$, which depend on $\mathbf{w}$. It then tries to adjust these potentials (via also adjusting $\mathbf{w}$) such that the minimizer $\hat{\mathbf{x}}^{(i,k)}$ of each slave MRF coincides with the desired solution $\bar{\mathbf{x}}^k$ on the nodes of G_i^k. Note that in such a case the resulting convex relaxation is tight (since the minimizers $\hat{\mathbf{x}}^{(i,k)}$ of all slave MRFs are consistent with each other) and thus the sum of hinge losses $\sum_i L_{G_i^k}(\cdot)$ equals the original hinge loss $L_{G^k}(\cdot)$. Of course, when adjusting the unary potentials of the slaves the algorithm must take into account that these are not independent but they have to satisfy constraint (23), *i.e.*, their sum should naturally equal the unary potentials of the original MRF on G^k.

At this point it is also worth looking at how the algorithm adjusts the unary potentials $\boldsymbol{\theta}^{(i,k)}$. This is done via updates (31), (25), both of which modify $\boldsymbol{\theta}^{(i,k)}$ due to (7). The aim of the former updates is to modify $\boldsymbol{\theta}^{(i,k)}$ such that the minimizers of different slave MRFs agree with each other. Indeed, it is easy to verify that the right hand side of (31) equals zero (*i.e.*, no update is applied to $\boldsymbol{\theta}^{(i,k)}$) only if all minimizers assign a common label to node p. On the contrary, if node p (contained, say, in only 2 sub-hypergraphs G_i^k, G_j^k) is assigned 2 different labels l_i, l_j during the t-th iteration (*i.e.*, $\hat{x}_p^{(i,k)} = l_i$, $\hat{x}_p^{(j,k)} = l_j$) then (31) reduces to the following update (assuming that $\alpha_t = 2\epsilon$)

$$\theta_p^{(i,k)}(l_i)\ {+}{=}\ \epsilon\ ,\ \ \theta_p^{(j,k)}(l_i)\ {-}{=}\ \epsilon\ , \tag{32}$$

$$\theta_p^{(i,k)}(l_j)\ {-}{=}\ \epsilon\ ,\ \ \theta_p^{(j,k)}(l_j)\ {+}{=}\ \epsilon\ , \tag{33}$$

which is easily seen to encourage a common label assignment for p in both G_i^k, G_j^k. Furthermore, the role of the second updates (25) is exactly to encourage this common label assignment to actually coincide with the desired label $\bar{x}_p^k$.

Regarding the correctness of the proposed algorithm the following general theorem follows directly from the fact that we make use of the projected subgradient method:

Theorem 1. *if multipliers $\alpha_t \geq 0$ satisfy $\lim_{t\to\infty}\alpha_t = 0$, $\sum_{t=0}^{\infty}\alpha_t = \infty$ then the proposed algorithm converges to an optimal solution of problem* (22)*.*

In addition, all known convergence rate results for subgradient methods carry over to our case.

6. Choice of decompositions $\{G_i^k\}$ and tighter approximations

A user of the proposed learning framework only needs to know how to compute a minimizer for a slave MRF. Everything else is being taken care of by the algorithm! As a result, the proposed framework provides extreme flexibility and generality. For instance, a user is allowed to use different decompositions $\{G_i^k\}$. This can improve the learning algorithm in various ways.

Let $\mathcal{F}_0$ denote the minimum of the original regularized loss function (16) and let $\mathcal{F}_{\{G_i^k\}}$ denote the minimum of loss function (22) that results when using decomposition $\{G_i^k\}$. The following holds true [1]:

Theorem 2. *Loss $\mathcal{F}_{\{G_i^k\}}$ upper bounds $\mathcal{F}_0$,* i.e.*, $\mathcal{F}_0{\leq}\mathcal{F}_{\{G_i^k\}}$*

Therefore, by minimizing $\mathcal{F}_{\{G_i^k\}}$ we are guaranteed to decrease the original loss $\mathcal{F}_0$ as well. However, by appropriately choosing the hypergraph decomposition $\{G_i^k\}$ we can improve the approximation $\mathcal{F}_{\{G_i^k\}}$ to the true loss $\mathcal{F}_0$. This happens because the tightness of the convex relaxation $\mathrm{DUAL}_{\{G_i^k\}}$ depends crucially on the choice of decomposition $\{G_i^k\}$. We will say decomposition $\{\tilde{G}_j^k\}$ is stronger than decomposition $\{G_i^k\}$ (and we will denote this by $\{G_i^k\} < \{\tilde{G}_j^k\}$) if the convex relaxation from $\{\tilde{G}_j^k\}$ is tighter than the relaxation from $\{G_i^k\}$, *i.e.*, it always holds $\mathrm{DUAL}_{\{G_i^k\}} {<} \mathrm{DUAL}_{\{\tilde{G}_j^k\}}$. The following theorem is true:

Theorem 3 ([1])**.** *If $\{G_i^k\}{<}\{\tilde{G}_j^k\}$ then $\mathcal{F}_0{\leq}\mathcal{F}_{\{\tilde{G}_j^k\}}{<}\mathcal{F}_{\{G_i^k\}}$,* i.e.*, $\mathcal{F}_{\{\tilde{G}_j^k\}}$ is a better approximation to $\mathcal{F}_0$ than $\mathcal{F}_{\{G_i^k\}}$.*

One decomposition we can choose is $G_{\mathrm{single}}^k = \{G_c^k\}$ that contains a sub-hypergraph $G_c^k = (\mathcal{V}_c^k, \mathcal{C}_c^k)$ for each clique $c \in \mathcal{C}^k$, where $\mathcal{V}_c^k = \{p | p \in c\}$ and $\mathcal{C}_c^k = \{c\}$, *i.e.*, each slave MRF consists of a single high-order clique. This is essentially the simplest possible slave, and is often very easy to solve regardless of the complexity of the original MRF. Therefore, the resulting learning algorithm can be used for training almost any high-order MRF. Furthermore, the convex relaxation resulting from G_{single}^k can be shown to coincide with the LP relaxation of the following integer

program [6]:

$$\min_{\mathbf{z}} \sum_p \sum_{x_p} \bar{u}_p^k(x_p) z_p(x_p) + \sum_c \sum_{\mathbf{x}_c} \bar{h}_c^k(\mathbf{x}_c) z_c(\mathbf{x}_c) \quad (34)$$

$$\text{s.t.} \sum_{x_p} z_p(x_p) = 1\,, \quad \forall p \quad (35)$$

$$\sum_{\mathbf{x}_c : x_p = l} z_c(\mathbf{x}_c) = z_p(l)\,, \quad \forall c \in \mathcal{C},\ p \in c \quad (36)$$

$$z_p(\cdot), z_c(\cdot) \in \{0,1\}\ , \quad (37)$$

where variables $z_p(x_p)$ and $z_c(\mathbf{x}_c)$ exist respectively for each label x_p of node p and each labeling $\mathbf{x}_c$ of clique c. The above relaxation generalizes the marginal polytope relaxation for pairwise MRFs, and often provides a good approximation to $\text{MRF}_{G^k}(\bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k)$.

However, one can also choose decompositions $\{\tilde{G}_j^k\}$ stronger than G_{single}^k, which thus lead to better approximations of loss $\mathcal{F}_0$. For instance, this can be achieved by taking advantage of the special structure that may exist in certain classes of MRFs. One such example was given in [6] for the case of MRFs with the so-called pattern-based potentials. More generally, the following theorem holds true:

Theorem 4 ([1]). *$\mathcal{F}_{\{\tilde{G}_j^k\}}$ is a better approximation to $\mathcal{F}_0$ than $\mathcal{F}_{G_{\text{single}}^k}$ only if at least one sub-hypergraph $\tilde{G}_j^k$ exists such that slave MRFs on $\tilde{G}_j^k$ do not have the integrality property*[2].

For instance, based on the above theorem one can provably derive a better learning algorithm for pairwise MRFs simply by using decompositions containing loopy subgraphs of small tree width (MRFs on such subgraphs can be efficiently optimized via the junction tree algorithm).

However, besides improving the accuracy of a learning algorithm, a proper choice of a decomposition $\{G_i^k\}$ can also improve the computational efficiency of that algorithm. For instance, in the case of pairwise MRFs a decomposition $G_{\text{tree}}^k = \{T_i^k\}$ consisting of spanning trees T_i^k may be used for that purpose. Although in this case the accuracy of learning is not improved compared to G_{single}^k (since $\text{DUAL}_{G_{\text{tree}}^k} = \text{DUAL}_{G_{\text{single}}^k}$ and thus $\mathcal{F}_{G_{\text{tree}}^k} = \mathcal{F}_{G_{\text{single}}^k}$), the speed of convergence does improve. The reason is because convex relaxation $\text{DUAL}_{G_{\text{tree}}^k}$ converges faster as each slave MRF now covers a much larger number of nodes. More generally, computational efficiency can be improved simply by choosing a decomposition that is specifically adapted to the class of MRFs we want to learn. For instance, if part of the energy of a MRF is known to be submodular we can take advantage of this fact simply by using that part as a slave. The very fast graph-cut based optimizers that exist for submodular energies can be used directly and will greatly reduce the computational cost of learning in this case.

[2]We say that an MRF has the integrality property if and only if the corresponding LP relaxation of (34) is tight.

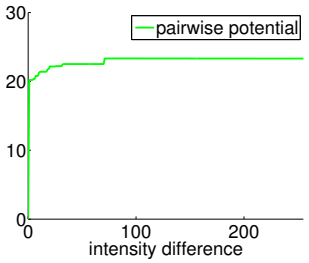
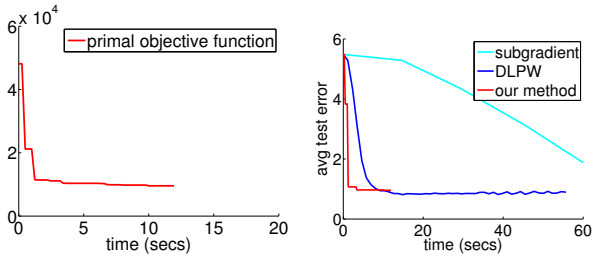


Fig. 2: Pairwise potential function $V(\cdot)$ learnt by our method.



(a) (b)

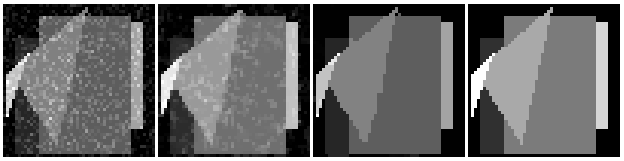
Fig. 3: **(a)** Primal objective (22) and **(b)** average test errors as a function of time.

7. Incremental and stochastic subgradient

To further improve computational efficiency we can use an incremental subgradient method, which is well suited to functions like (22) that can be expressed as a sum of component functions. According to this method at each iteration we need to take a step along the subgradient of only one component function, where this component function is picked either deterministically (by repeatedly visiting all component functions in a fixed order) or uniformly at random. A component function for objective function (22) can have the following form: $\mu' R(\cdot) + \sum_{i \in S} L_{G_i^k}(\cdot)$. This means that at each iteration we consider the hinge losses $L_{G_i^k}(\cdot)$ for only a subset of slave MRFs from a selected training sample of index k (in this case updates are similar to (31), (25) but they take into account only a subset of slaves). In a randomized version we first pick the index of a training sample k randomly from $\{1, \ldots, K\}$, and then the subset S is picked also randomly from a predefined partition of the slave indices of the k-th sample. If S is always chosen to contain all slave indices of the k-th sample then this is essentially equivalent to the more well known stochastic subgradient algorithm. Just like the subgradient method, incremental subgradient is guaranteed to converge to an optimal solution since a similar theorem to Thm. 1 holds true in this case [12].

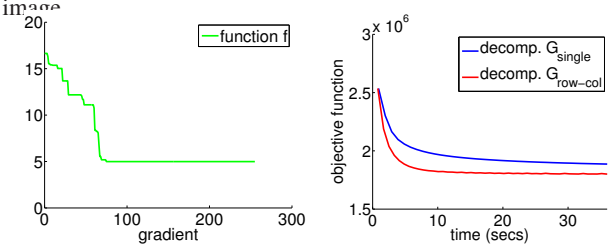
8. Experimental results

We next evaluate our algorithms using various experiments. The first experiment is about image denoising. In this case, we have created training and testing datasets con-



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

Fig. 4: **(a)** Noisy test image **(b)** Denoised image when using a function $f(\cdot)$ estimated during the course of the learning algorithm **(c)** Denoised result when using the final $f(\cdot)$ **(d)** Ground truth image



(a) (b)

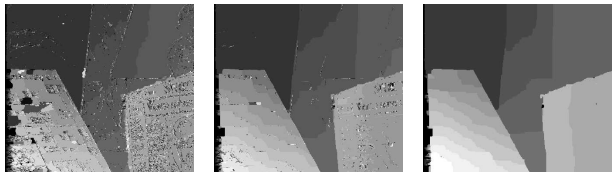
Fig. 5: **(a)** Learnt function $f(\cdot)$ **(b)** Primal objective (22) as a function of time for two different graph decompositions.

sisting of synthetic piecewise constant images that have been corrupted by gaussian noise (see Fig. 4). To denoise these images we will use a pairwise MRF whose unary potential will be $u_p^k(l) = |l - I_p|$, where I_p denotes image intensity at pixel p. We also assume that the MRF pairwise potential $h_{pq}^k(\cdot,\cdot)$ has the following form $h_{pq}^k(l_p, l_q) = V(|l_p - l_q|)$ Our goal is to learn the underlying function $V(\cdot)$, which means that we need to estimate a vector $\mathbf{w}$ of size 256, each component of which corresponds to one value of $V(\cdot)$. The resulting $V(\cdot)$ after applying our method on a training set of 10 images is shown in Fig. 2. Although $V(\cdot)$ was determined automatically, it looks very much like a truncated linear function, which fully agrees with the common practice of using this type of discontinuity preserving potentials when dealing with piecewise constant images. Fig. 3(a) shows how the primal objective function (22) varies during the course of our algorithm. Notice how quickly convergence takes place. We also compare to two other methods: the subgradient algorithm from [13] and the DLPW algorithm from [10]. Fig. 3(b) shows the average test error (for a test set of 10 noisy synthetic images) as a function of time for each algorithm. Our method manages to reduce the test error faster than DLPW. Similarly, it is a lot more efficient than the subgradient method [13]. The inefficiency of the method [13] comes from the fact that the computation of a subgradient is much more expensive than in our case since it requires fully solving a LP-relaxation for problem $\mathrm{MRF}_{G^k}(\bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k)$ (*i.e.*, for a MRF defined on the whole graph). We also show in Fig. 4 a sample result produced when denoising a test image using the function $V(\cdot)$ learnt by our method.

We next test our method on an application for stereo matching. In this case we want to estimate a disparity per pixel and we will use a pairwise MRF with unary potential given by $u_p^k(l) = |I_p^{\text{left}} - I_{p-l}^{\text{right}}|$, where I^{left}, I^{right} are the left and right images. A very commonly used pairwise potential in this case is a gradient-modulated Potts model of the following form: $h_{pq}^k(l_p, l_q) = f(|\nabla I_p^{\text{left}}|)[l_p \neq l_q]$, where $\nabla I_p^{\text{left}} = I_p^{\text{left}} - I_q^{\text{left}}$ represents the gradient of the left image at p. Our goal is to automatically learn the function $f(\cdot)$ that is used for assigning a discontinuity penalty depending on the magnitude of that gradient. Function $f(\cdot)$ can take 256 different values assuming integer intensities and so the positive vector $\mathbf{w}$ that we need to estimate will be of size 256 with $w_i = f(i)$. In this case we also impose the restriction that vector $\mathbf{w}$ should belong to set $\mathcal{W}$, where $\mathcal{W} = \{\mathbf{w} \geq 0 | w_i \geq w_{i+1}\}$, thus reflecting the a priori knowledge that $f(\cdot)$ should be a decreasing function. In this case the projected subgradient algorithm simply requires applying an additional projection step $\mathbf{w} \leftarrow \mathrm{Proj}_{\mathcal{W}}(\mathbf{w})$ at the end of each iteration, which is the only modification needed by our method. We show in Fig. 5(a) the resulting function $f(\cdot)$ that was estimated by our learning algorithm using as training set two stereo pairs from the middlebury stereo dataset (the 'Tsukuba' and the 'Map' pairs were used). Using this function, we computed disparity maps for the 'Venus', 'Sawtooth', 'Bull' and 'Poster' stereo pairs from the middlebury dataset (see Fig. 6, Fig. 7). The corresponding disparity error rates were 4.9%, 4.4%, 2.8%, 3.7% respectively. Fig. 7 also shows 3 different disparity maps that were computed for one of these test images using the function $f(\cdot)$ as estimated at 3 different iterations of our learning algorithm. Notice how the errors in the disparity are reduced as the algorithm converges. Fig. 5(b) shows how the primal objective (22) varies as a function of time

(a) (b) (c)

Fig. 6: Disparity maps for the 'Sawtooth', 'Poster' and 'Bull' stereo pairs.



(a) (b) (c)

Fig. 7: Three disparity maps computed for the stereo pair 'Venus' using functions $f(\cdot)$ estimated at different iterations of our learning algorithm (the final result is the one shown in (c)).

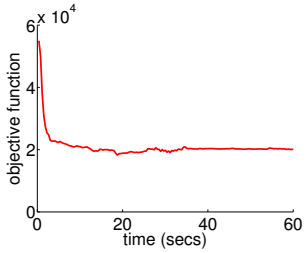


Fig. 8: Primal objective function during training of high-order MRFs.

during learning. Notice again that our method manages to successfully reduce this objective function very fast. On the contrary, the subgradient method [13] is not practical to use in this case due to the large size of the MRF problems, which increases the running time considerably.

We also compare in Fig. 5(b) the effect of using two different decompositions in our learning algorithm. Decomposition $G_{\text{row-col}}$ uses each row or column of the MRF grid as a slave, while G_{single} uses only one edge in the MRF graph per slave. As explained in §6, $G_{\text{row-col}}$ is expected to converge faster, which is indeed what happens here.

Finally, to demonstrate the generality and flexibility of our approach we also did some synthetic experiments on learning high order MRFs. More specifically, we applied our method to MRFs with $\mathcal{P}^n$ Potts high-order potentials, which have the following form:

$$h_c(\mathbf{x}) = \begin{cases} \beta_l^c & \text{if } x_p = l,\ \forall p \in c \\ \beta_{\max}^c & \text{otherwise} \ , \end{cases} \tag{38}$$

where l denotes any label from a discrete set of labels $\mathcal{L}$. We assume that each β_l^c is equal to the dot product of a vector of parameters $\mathbf{w}_l$ with a feature vector z_l^c, *i.e.*, $\beta_l^c = \mathbf{w}_l \cdot z_l^c$, and the goal of learning includes estimating all vectors $\mathbf{w}_l$. For this we use synthetic data: we randomly sample unary potentials as well as feature vectors $\{z_l^c\}$ and then we generate the values β_l^c of the high-order potentials based on a specified set of vectors $\{\mathbf{w}_l\}$. We then approximately minimize the resulting MRF using the method from [6], and the solution $\bar{\mathbf{x}}$ that we obtain is used as the ground truth for the current sample (we repeat this process to generate as many samples as we want). For the corresponding MRF hypergraph we assume that its nodes are arranged in a 2D grid and there exists a high-order clique for each subrectangle of size $s \times s$ in that grid. Our learning algorithm can be applied to this case by using a decomposition that assigns one clique per slave. Note that the minimization of each slave takes time $O(|\mathcal{L}|)$ and thus can be achieved very efficiently regardless of the size of the high-order clique. Fig. 8 shows an example of how fast the primal objective function (22) decreases during learning in this case (we used a grid of size 50×50, the clique size was 3×3, $|\mathcal{L}| = 5$, and we used 100 training samples). The main point we want to emphasize here is the efficiency of our method even in the case of training a high-order MRF.

Before finishing we note that the function $\Delta(\cdot, \cdot)$ in (12) was set equal to the hamming loss for all the experiments in the paper.

9. Conclusions

We have presented an algorithmic framework that can be used for training arbitrary MRFs/CRFs. It essentially manages to reduce the training of a complex high-order MRF to that of training a set of simple random field models. The derived learning scheme is very general, highly efficient and extremely flexible (*e.g.*, it can be applied to both pairwise and high-order models, it requires no submodularity assumptions, it is easily adapted to the structure of a given class of MRFs *etc.*). Due to these properties, we thus believe that our framework will find use in a broad class of computer vision problems, especially now where learning problems are becoming increasingly important as well as challenging for a great variety of applications.

References

[1] supplemental material.
[2] K. Alahari, C. Russell, and P. Torr. Efficient piecewise learning for conditional random fields. In *CVPR*, 2010.
[3] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of markov random fields for segmentation of 3d scan data. In *CVPR*, 2005.
[4] T. Finley and T. Joachims. Training structural svms when exact inference is intractable. In *ICML*, 2008.
[5] V. Franc and B. Savchynskyy. Discriminative learning of max-sum classifiers. *JMLR*, 2008.
[6] N. Komodakis and N. Paragios. Beyond pairwise energies: Efficient optimization for higher-order MRFs. In *CVPR*, 2009.
[7] N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *ICCV*, 2007.
[8] N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *PAMI*, 2010.
[9] S. Kumar, J. August, and M. Hebert. Exploiting inference for approximate parameter learning in discriminative fields: An empirical study. In *EMMCVPR*, 2005.
[10] O. Meshi, D. Sontag, T. Jaakkola, and A. Globerson. Learning efficiently with approximate inference via dual losses. In *ICML*, 2010.
[11] D. Munoz, J. A. D. Bagnell, N. Vandapel, and M. Hebert. Contextual classification with functional max-margin markov networks. In *CVPR*, 2009.
[12] A. Nedic and D. P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM J. on Optimization*, 2001.
[13] N. Ratliff, J. A. D. Bagnell, and M. Zinkevich. (online) subgradient methods for structured prediction. In *AIStats*, 2007.
[14] D. Scharstein and C. Pal. Learning conditional random fields for stereo. In *CVPR*, 2007.
[15] M. Szummer, P. Kohli, and D. Hoiem. Learning CRFs using graph cuts. In *ECCV*, 2008.
[16] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *NIPS*, 2004.