Learning to Cluster Using High Order Graphical Models with Latent Variables

Nikos Komodakis
University of Crete, Computer Science Department
http://www.csd.uoc.gr/~komod

Abstract

This paper proposes a very general max-margin learning framework for distance-based clustering. To this end, it formulates clustering as a high order energy minimization problem with latent variables, and applies a dual decomposition approach for training this model. The resulting framework allows learning a very broad class of distance functions, permits an automatic determination of the number of clusters during testing, and is also very efficient. As an additional contribution, we show how our method can be generalized to handle the training of a very broad class of important models in computer vision: arbitrary high-order latent CRFs. Experimental results verify its effectiveness.

1. Introduction

Clustering is known to be ubiquitous in computer vision, playing a crucial role in a wide variety of applications. It is typically cast as an optimization problem, where one aims to estimate a set of cluster centers that optimally represent a given dataset based on a distance function measuring dissimilarity between data points. The choice of this distance plays a crucial role for the success of the method. Due to the complexity and variability of the clustering tasks encountered in computer vision, the ability to automatically *learn* such a distance based on training data is a matter of utmost importance. In fact, quite often this is the only viable choice as the distances to be learnt may depend on a large number of parameters.

Considering the above observations, the goal of this work is to provide a very general max-margin learning framework for distance-based clustering. It uses as input a set of ground truth partitions of training datasets. Based just on this input, it enables the discriminative learning of a very broad class of distances for clustering, where, *e.g.*, non-metric, non-differentiable or even asymmetric distances can be handled by our method. Furthermore, despite its generality, the proposed learning framework provides great computational efficiency as it is based on a very fast and parallelizable projected subgradient method. On top of that, it relies on a formulation of clustering that allows the number of extracted clusters to be automatically determined as a result of the optimization process. Therefore, our method is able to properly account for the fact that this number is typically not known in advance at test time (note that a correct estimation of this number often proves a very crucial factor).

To achieve all these, we formulate the learning problem for clustering as one of training a discrete conditional random field (CRF). The two complications that we must deal with in this case are the following: the resulting CRF is of very high order, and it also contains latent variables that are not observable during training. As a result, we also propose a very general method for handling learning problems of this type, which is another important contribution of this work. The proposed method relies on a master-slave dual decomposition approach [10, 8] thanks to which it manages to reduce the max-margin training of a complex high order CRF with *latent* variables to that of training a series of much simpler slave CRFs. This gives to the proposed method great generality and flexibility, thus allowing it to efficiently handle a very broad class of high-order latent CRF models.

Regarding prior work, a fully supervised learning method for clustering has also been proposed in [4]. However, that method relies on a different formulation based on correlation clustering. Such a formulation requires the use of a similarity function sim_{pq} that classifies each pair of datapoints (p, q) as either similar $(\text{sim}_{pq} > 0)$ or dissimilar $(\text{sim}_{pq} < 0)$. Also, a structured learning method for models with latent variables has appeared in [17]. However, both [17] and [4] require solving a very complicated LP relaxation as well as a quadratic program with a growing number of constraints per iteration. They are thus much slower than our approach. Furthermore, they can only handle quadratic regularizers. On the contrary, our method can naturally handle any type of regularizer, such as a sparsity-inducing l_1 norm that often plays a very crucial role for correct learning. Our method also extends the recently proposed learning framework [6]. Although that framework can efficiently train arbitrary high-order CRFs, it cannot handle any hidden variables during learning. Before proceeding, we should also note the important role that latent CRFs play in many vision applications [3].

Paper structure: §2 describes the used formulation of clustering. The corresponding max-margin learning framework

is presented in §3-§4, while §5 explains how it can be generalized to arbitrary high-order latent CRFs. Experimental results are presented in §6, and we finally conclude in §7.

2. Exemplar based clustering

For a set of datapoints S endowed with a distance, a general formulation for center based clustering is the following one [9, 5]

$$\min_{Q \subseteq S} E(Q) = \sum_{p \notin Q} \min_{q \in Q} d_{p,q} + \sum_{q \in Q} d_{q,q} \ . \tag{1}$$

Here Q represents the set of cluster centers (exemplars), which in this case can consist of any subset of data points from the input set S. The role of elements of $\mathbf{d} = \{d_{p,q}\}$ is twofold: for $p \neq q$ each $d_{p,q}$ represents the distance between p and q, whereas each element $d_{q,q}$ represents the penalty for choosing q as exemplar. As a result, we seek to minimize the distance of a datapoint to its nearest center, while at the same time choosing as few centers as possible. Note that in this case the number of exemplars is not predetermined but is an output of the optimization. In addition, this formulation allows us to use very general distances.

The above formulation of clustering can be cast as a high-order CRF optimization problem [9]. To this end, note that problem (1) is equivalent to the following integer program

$$\min_{\mathbf{x}} \sum_{p,q \in S} d_{p,q} x_{pq} \tag{2}$$

$$\text{s.t.} \sum_{q \in S} x_{pq} = 1, \ \ \forall p \tag{3}$$

$$x_{pq} \leq x_{qq}, \qquad \forall p, q \tag{4}$$

$$x_{pq} \in \{0, 1\}, \ \ \forall p, q. \tag{5}$$

In this integer program each binary variable x_{qq} indicates whether q has been chosen as an exemplar or not, while x_{pq} with $p \neq q$ indicates whether p has been assigned to exemplar q or not. Constraints (3) ensure that each p is assigned to exactly one cluster, while constraints (4) ensure that if p is assigned to q then the latter must be an exemplar.

Therefore clustering can be expressed as minimizing the following function

$$E(\mathbf{x}; \mathbf{d}) = \sum_{p,q} u_{pq}(x_{pq}; \mathbf{d}) + \sum_{p,q} \phi_{pq}(x_{pq}, x_{qq}) + \sum_{p} \phi_p(\mathbf{x}_p) \tag{6}$$

which represents the energy of a discrete CRF having unary potentials $\mathbf{u} = \{u_{pq}(\,\cdot\,; \mathbf{d})\}$ and higher order potentials $\boldsymbol{\phi} = \{\phi_{pq}(\cdot), \phi_p(\cdot)\}$ that are defined as

$$u_{pq}(x_{pq}; \mathbf{d}) = d_{p,q} x_{pq} \tag{7}$$

$$\phi_{pq}(x_{pq}, x_{qq}) = \delta(x_{pq} \leq x_{qq}) \tag{8}$$

$$\phi_p(\mathbf{x}_p) = \delta\left(\sum\nolimits_q x_{pq} = 1\right), \tag{9}$$

where $\mathbf{x}_p = \{x_{pq} | q \in S\}$, and $\delta(\cdot)$ equals 0 if the expression in parenthesis is satisfied and ∞ otherwise.

3. Max-margin learning for clustering

Let us now assume that we are given a set of K training samples $\{S^k, \mathcal{C}^k, \mathbf{y}^k\}_{k=1}^K$. Here S^k represents the set of data points of the k-th training sample, and $\mathcal{C}^k = \{C_i^k\}$ represents a ground truth partition (clustering) of that set, *i.e.*, it holds $\cup_i C_i^k = S^k$, $C_i^k \cap C_j^k = \emptyset, \forall i \neq j$. We also assume that the distances between data points of the k-th sample $\mathbf{d}^k = \{d_{p,q}^k\}$ can be expressed in terms of an unknown vector of parameters $\mathbf{w}$ and a set of known vector-valued feature functions $f_{pq}(\cdot)$ that depend on some input data $\mathbf{y}^k$, *i.e.* it holds

$$d_{p,q}^k = \mathbf{w}^T f_{pq}(\mathbf{y}^k) \ . \tag{10}$$

We seek to estimate $\mathbf{w}$ such that when we minimize the energy $E(\mathbf{x}; \mathbf{d})$ resulting from any given input data $\mathbf{y}$, we can then predict the correct clustering $\mathcal{C}$ for the corresponding set of data points S. However, there are two difficulties associated with this structured learning task that we need to deal with in this case: on the one hand, the energy $E(\mathbf{x}; \mathbf{d})$ is of very high-order (*e.g.*, notice that potential $\phi_p(\mathbf{x}_p)$ is a function of $|S|$ variables). On the other hand, during training we get to observe only the correct partition $\mathcal{C}^k$ but not the underlying vector $\mathbf{x}^k$, *i.e.*, variables $\mathbf{x}^k$ are latent in this case (note that a partition $\mathcal{C}^k$ does not fully determine $\mathbf{x}^k$ since the cluster centers are unknown). Hereafter we will denote by $\mathcal{X}(\mathcal{C})$ the set of all $\mathbf{x}$ consistent[1] with the clustering defined by partition $\mathcal{C}$, and we will also use the following notation throughout the rest of the paper:

$$E^k(\mathbf{x}; \mathbf{w}) := E(\mathbf{x}; \mathbf{d}^\mathbf{k}) \ , \ \ u_{pq}^k(x_{pq}) := u_{pq}(x_{pq}; \mathbf{d}^k) \tag{11}$$

To estimate $\mathbf{w}$, here we will follow a max-margin approach [14], in which case we must adjust $\mathbf{w}$ such that there exists $\mathbf{x}^k \in \mathcal{X}(\mathcal{C}^k)$ whose energy $E^k(\mathbf{x}^k; \mathbf{w})$ is smaller by $\Delta(\mathbf{x}; \mathcal{C}^k)$ than the energy of any other binary solution $\mathbf{x}$, *i.e.*

$$\exists \mathbf{x}^k \in \mathcal{X}(\mathcal{C}^k) : E^k(\mathbf{x}^k; \mathbf{w}) \leq E^k(\mathbf{x}; \mathbf{w}) - \Delta(\mathbf{x}; \mathcal{C}^k) + \xi_k \ . \tag{12}$$

Function $\Delta(\mathbf{x}; \mathcal{C}^k)$ should measure how far the clustering induced by $\mathbf{x}$ is with respect to any desired clustering $\mathcal{C}^k$. Furthermore, slack variable ξ_k is required for the case of an infeasible training set. Constraints (12) lead to minimizing the following regularized loss functional for computing $\mathbf{w}$

$$\min_{\{\mathbf{x}^k \in \mathcal{X}(\mathcal{C}^k)\}, \mathbf{w}} \tau J(\mathbf{w}) + \sum\nolimits_k \mathcal{L}_{E^k}(\mathbf{x}^k; \mathbf{w}) \ , \tag{13}$$

where $J(\mathbf{w})$ is a regularization term, while $\mathcal{L}_{E^k}(\mathbf{x}^k; \mathbf{w})$ represents slack variable ξ_k and is given by the following

[1] Vector $\mathbf{x}$ is said to be consistent with clustering $\mathcal{C}$ iff the partition induced by $\mathbf{x}$ coincides with $\mathcal{C}$.

hinge loss

$$\mathcal{L}_{E^k}(\mathbf{x}^k;\mathbf{w}) = E^k(\mathbf{x}^k;\mathbf{w}) - \min_{\mathbf{x}} \left(E^k(\mathbf{x};\mathbf{w}) - \Delta(\mathbf{x};\mathcal{C}^k)\right). \tag{14}$$

The selection of $J(\mathbf{w})$ often proves a crucial factor depending on the learning task. Many possible choices exist for $J(\mathbf{w})$ (*e.g.*, $||\mathbf{w}||^2$, a sparsity-inducing norm $||w||_1$ *etc.*).

3.1. Choosing the error function $\Delta(\mathbf{x};\mathcal{C}^k)$

The proper definition of function $\Delta(\mathbf{x};\mathcal{C}^k)$ is important for learning a correct $\mathbf{w}$. As mentioned above, $\Delta(\mathbf{x};\mathcal{C}^k)$ must provide a measure of the error between the clustering induced by $\mathbf{x}$ and the clustering induced by partition $\mathcal{C}^k$. To this end, we will make use of the following definition

$$\Delta(\mathbf{x};\mathcal{C}^k) = \alpha \sum_{C\in\mathcal{C}^k} \Big|1 - \sum_{q\in C} x_{qq}\Big| + \beta \sum_{C\in\mathcal{C}^k}\sum_{p\in C} \Big(1 - \sum_{q\in C} x_{pq}\Big). \tag{15}$$

Notice that the first term equals zero if and only if for each cluster $C \in \mathcal{C}^k$ there exists exactly one datapoint chosen as exemplar by $\mathbf{x}$ among all datapoints in C. Furthermore, the second term is zero if and only if this unique exemplar for cluster C, say q', is assigned by solution $\mathbf{x}$ to all datapoints p in C, *i.e.* it holds $x_{pq'} = 1, \forall p \in C$ (the constants $\alpha, \beta \geq 0$ are used for weighting the importance of these two terms). Therefore, the following natural property holds true

$$\Delta(\mathbf{x};\mathcal{C}^k) = 0 \Leftrightarrow \mathbf{x} \in \mathcal{X}(\mathcal{C}^k). \tag{16}$$

In addition, it is easy to see that the more inconsistent solution $\mathbf{x}$ is with respect to clustering $\mathcal{C}^k$, the greater the value of the error $\Delta(\mathbf{x};\mathcal{C}^k)$ is.

Let us now define $\bar{E}^k(\mathbf{x};\mathbf{w}) := E^k(\mathbf{x};\mathbf{w}) - \Delta(\mathbf{x};\mathcal{C}^k)$. An important observation for the analysis that will follow is that $\bar{E}^k(\mathbf{x};\mathbf{w})$ can still be expressed as the energy of a high-order discrete CRF, *i.e.* it holds

$$\begin{aligned}\bar{E}^k(\mathbf{x};\mathbf{w}) = &\sum_{p,q} \bar{u}^k_{pq}(x_{pq}) + \sum_{p,q} \bar{\phi}_{pq}(x_{pq}, x_{qq}) + \\ &\sum_{p} \bar{\phi}_p(\mathbf{x}_p) + \sum_{C\in\mathcal{C}^k} \bar{\phi}_C(\mathbf{x}_C) - \beta|S^k| \ ,\end{aligned} \tag{17}$$

where $\mathbf{x}_C = \{x_{qq} \,|\, q \in C\}$. The unary potentials $\bar{\mathbf{u}}^k = \{\bar{u}^k_{pq}\}$ and the higher-order potentials $\bar{\boldsymbol{\phi}} = \{\bar{\phi}_{pq}, \bar{\phi}_p, \bar{\phi}_C\}$ equal

$$\bar{u}^k_{pq}(x_{pq}) = u^k_{pq}(x_{pq}) + \beta \cdot \big[\exists C \in \mathcal{C}^k : p, q \in C\big] \cdot x_{pq}, \tag{18}$$

$$\bar{\phi}_{pq}(\cdot) = \phi_{pq}(\cdot) \ \ , \ \ \bar{\phi}_p(\cdot) = \phi_p(\cdot) \ , \tag{19}$$

$$\bar{\phi}_C(\mathbf{x}_C) = -\alpha\Big|1 - \sum\nolimits_{q\in C} x_{qq}\Big| \ , \tag{20}$$

where $[\,\cdot\,]$ denotes the indicator function (which is 1 if the expression in square brackets is true, and 0 otherwise).

Due to (16) and the fact that $\mathbf{x}^k \in \mathcal{X}(\mathcal{C}^k)$ it follows that $\bar{E}^k(\mathbf{x}^k;\mathbf{w}) = E^k(\mathbf{x}^k;\mathbf{w})$. Therefore, the following equality holds $\mathcal{L}_{E^k}(\mathbf{x}^k;\mathbf{w}) = \bar{\mathcal{L}}_{\bar{E}^k}(\mathbf{x}^k;\mathbf{w})$, where we define

$$\bar{\mathcal{L}}_{\bar{E}^k}(\mathbf{x}^k;\mathbf{w}) := \bar{E}^k(\mathbf{x}^k;\mathbf{w}) - \min_{\mathbf{x}} \bar{E}^k(\mathbf{x};\mathbf{w}) \ . \tag{21}$$

As a result, loss function (13) reduces to

$$\min_{\{\mathbf{x}^k\in\mathcal{X}(\mathcal{C}^k)\},\mathbf{w}} \tau J(\mathbf{w}) + \sum\nolimits_k \bar{\mathcal{L}}_{\bar{E}^k}(\mathbf{x}^k;\mathbf{w}) \ . \tag{22}$$

Essentially the above loss tells us the following fact: during learning we should ideally adjust $\mathbf{w}$ such that the minimum of the high-order energy $\bar{E}^k(\,\cdot\,;\mathbf{w})$ is attained at a solution $\mathbf{x}^k$ that belongs to the set $\mathcal{X}(\mathcal{C}^k)$.

4. Learning to cluster using a dual decomposition approach

Dealing with the above loss functional is intractable. This happens due to the term $\min_{\mathbf{x}} \bar{E}^k(\mathbf{x};\mathbf{w})$ appearing in $\bar{\mathcal{L}}_{\bar{E}^k}(\mathbf{x}^k;\mathbf{w})$, which is NP-hard to compute. In cases like this, one typically opts to approximate the original loss with an easier to handle upper bound, which should be minimized so as to implicitly lead to a reduction of the original loss as well. Having expressed the clustering-related function $\bar{E}^k(\,\cdot\,;\mathbf{w})$ as the energy of a high-order CRF, here we will derive such an upper bound by approximating $\min_{\mathbf{x}} \bar{E}^k(\mathbf{x};\mathbf{w})$ with a convex dual relaxation that results from applying the method of dual decomposition to $\bar{E}^k(\,\cdot\,;\mathbf{w})$ [7, 10]. According to this method, the energy $\bar{E}^k(\,\cdot\,;\mathbf{w})$ (also called the energy of the *master* problem) must be decomposed into the energies of a set of *slave* subproblems. In this case, we will employ the following specific decomposition $\big\{\{\bar{E}^k_p\}_{p\in S^k}, \{\bar{E}^k_C\}_{C\in\mathcal{C}^k}\big\}$, which makes use of one slave subproblem per datapoint $p \in S^k$ and one slave subproblem per cluster $C \in \mathcal{C}^k$, where

$$\begin{aligned}\bar{E}^k_p(\mathbf{x};\mathbf{w},\boldsymbol{\lambda}) = &\sum_{q:q\neq p} \bar{u}^k_{pq}(x_{pq}) + \sum_{q} \bar{\phi}_{pq}(x_{pq}, x_{qq}) + \bar{\phi}_p(\mathbf{x}_p) \\ &+ \sum_{q} \Big(\frac{\bar{u}^k_{qq}(x_{qq})}{|S^k|+1} + \lambda_{pq} x_{qq}\Big) - \beta \ ,\end{aligned} \tag{23}$$

$$\bar{E}^k_C(\mathbf{x};\mathbf{w},\boldsymbol{\lambda}) = \sum_{q\in C} \Big(\frac{\bar{u}^k_{qq}(x_{qq})}{|S^k|+1} + \lambda_{Cq} x_{qq}\Big) + \bar{\phi}_C(\mathbf{x}_C) \, . \tag{24}$$

Here, the vector $\boldsymbol{\lambda} = \big\{\{\lambda_{pq}\}, \{\lambda_{Cq}\}\big\}$ (whose components represent the dual variables) is assumed to satisfy

$$\boldsymbol{\lambda} \in \boldsymbol{\Lambda}^k = \big\{\boldsymbol{\lambda} : \sum_{p\in S^k} \lambda_{pq} + \lambda_{Cq} = 0 \, , \ \forall C \in \mathcal{C}^k, q \in C\big\} \, . \tag{25}$$

Due to (25), it holds $\bar{E}^k = \sum_p \bar{E}^k_p + \sum_C \bar{E}^k_C$ and so the sum of the minimum energies of the slave subproblems is easily shown to provide a lower bound to the minimum energy of the master problem, *i.e.*, $\sum_p \min_{\mathbf{x}} \bar{E}^k_p(\mathbf{x};\mathbf{w},\boldsymbol{\lambda}) + \sum_C \min_{\mathbf{x}} \bar{E}^k_C(\mathbf{x};\mathbf{w},\boldsymbol{\lambda}) \leq \min_{\mathbf{x}} \bar{E}^k(\mathbf{x};\mathbf{w})$. Maximizing this lower bound by adjusting $\boldsymbol{\lambda}$ leads to the convex dual

relaxation $\mathcal{R}^k(\mathbf{w})$ used for approximating $\min_{\mathbf{x}} \bar{E}^k(\mathbf{x};\mathbf{w})$

$$\mathcal{R}^k(\mathbf{w}) = \max_{\boldsymbol{\lambda}\in\boldsymbol{\Lambda}^k} \Big(\sum_p \min_{\mathbf{x}} \bar{E}^k_p(\mathbf{x};\mathbf{w},\boldsymbol{\lambda}) + \sum_C \min_{\mathbf{x}} \bar{E}^k_C(\mathbf{x};\mathbf{w},\boldsymbol{\lambda}) \Big) \tag{26}$$

Therefore, the loss function that we must finally minimize results from replacing $\min_{\mathbf{x}} \bar{E}^k(\mathbf{x};\mathbf{w})$ with $\mathcal{R}^k(\mathbf{w})$ in (22), thus leading to the following upper bound of the true loss

$$\min_{\{\mathbf{x}^k\in\mathcal{X}(\mathcal{C}^k)\},\mathbf{w}} \tau J(\mathbf{w}) + \sum_k \big(\bar{E}^k(\mathbf{x}^k;\mathbf{w}) - \mathcal{R}^k(\mathbf{w}) \big). \tag{27}$$

By substituting (26) for $\mathcal{R}^k(\mathbf{w})$, the minimization (27) above can be shown [1] to finally reduce to

$$\min_{\{\mathbf{x}^k\in\mathcal{X}(\mathcal{C}^k)\},\mathbf{w},\{\boldsymbol{\lambda}^k\in\boldsymbol{\Lambda}^k\}} \tau J(\mathbf{w}) + \sum_k \sum_{p\in S^k} \bar{\mathcal{L}}_{\bar{E}^k_p}(\mathbf{x}^k;\mathbf{w},\boldsymbol{\lambda}^k) + \sum_k \sum_{C\in\mathcal{C}^k} \bar{\mathcal{L}}_{\bar{E}^k_C}(\mathbf{x}^k;\mathbf{w},\boldsymbol{\lambda}^k), \tag{28}$$

where $\bar{\mathcal{L}}_{\bar{E}^k_p}$, $\bar{\mathcal{L}}_{\bar{E}^k_C}$ are defined analogously to (21) (*i.e.*, we replace $\bar{E}^k(\cdot\,;\mathbf{w})$ in (21) with $\bar{E}^k_p(\cdot\,;\mathbf{w},\boldsymbol{\lambda}^k)$ and $\bar{E}^k_C(\cdot\,;\mathbf{w},\boldsymbol{\lambda}^k)$ respectively).

Therefore, the initial loss function (22), which was intractable due to including the hinge losses $\bar{\mathcal{L}}_{\bar{E}^k}(\cdot)$ of the extremely complicated energy functions $\bar{E}^k$, has now been substituted with the loss function (28) that involves hinge losses $\bar{\mathcal{L}}_{\bar{E}^k_p}(\cdot)$, $\bar{\mathcal{L}}_{\bar{E}^k_C}(\cdot)$ related to the much easier energies of the slave subproblems $\bar{E}^k_p, \bar{E}^k_C$. As we shall see below, this will lead to an extremely efficient learning scheme for clustering. Note that the function in (28) still remains non-convex (*e.g.*, due to the terms $\bar{E}^k_p(\mathbf{x}^k;\mathbf{w},\boldsymbol{\lambda}^k)$ in $\bar{\mathcal{L}}_{\bar{E}^k_p}$). To minimize it we will use a block-coordinate descent approach by alternately optimizing over $\{\mathbf{x}^k\}$ and $\{\mathbf{w},\{\boldsymbol{\lambda}^k\}\}$.

4.1. Optimizing over $\{\mathbf{x}^k\}$

Minimizing function (28) over $\{\mathbf{x}^k\}$ (for fixed $\mathbf{w}$) gives

$$\mathbf{x}^k = \arg\min_{\mathbf{x}\in\mathcal{X}(\mathcal{C}^k)} \bar{E}^k(\mathbf{x};\mathbf{w}) \overset{(16)}{=} \arg\min_{\mathbf{x}\in\mathcal{X}(\mathcal{C}^k)} E^k(\mathbf{x};\mathbf{w}).$$

Therefore, to fully determine $\mathbf{x}^k$ it suffices to find the set of exemplars Q^k that are consistent with $\mathcal{C}^k$ and also minimize the clustering cost $E^k(\mathbf{x};\mathbf{w})$ (indeed, if we know Q^k then we can set $x^k_{qq} = 1 \Leftrightarrow q \in Q^k$, while for each $p \neq q$ we can assign $x^k_{pq} = 1 \Leftrightarrow q = \arg\min_{q\in Q^k} d^k_{p,q}$). The computation of the set of exemplars Q^k can be performed in linear time as follows: since Q^k must be consistent with $\mathcal{C}^k$, it must have the following form $Q^k = \{q_C\}_{C\in\mathcal{C}^k}$, where q_C denotes the unique exemplar corresponding to cluster C. Furthermore, this unique exemplar q_C can be efficiently found via the following minimization

$$q_C = \arg\min_{q\in C} \sum_{p\in C} d^k_{p,q} \ . \tag{29}$$

4.2. Optimizing over $\{\mathbf{w},\{\boldsymbol{\lambda}^k\}\}$

When $\{\mathbf{x}^k\}$ are fixed, each term $\bar{\mathcal{L}}_{\bar{E}^k_p}(\mathbf{x}^k;\mathbf{w},\boldsymbol{\lambda}^k)$ (resp. $\bar{\mathcal{L}}_{\bar{E}^k_C}(\mathbf{x}^k;\mathbf{w},\boldsymbol{\lambda}^k)$) is immediately recognized to correspond to the structured learning loss of a slave CRF with energy $\bar{E}^k_p$ (resp. $\bar{E}^k_C$), where output variables $\mathbf{x}^k$ are now assumed to be fully observed during training (*i.e.*, no latent variables exist). As a result, for fixed $\{\mathbf{x}^k\}$, optimizing convex function (28) essentially has been reduced to the parallel training of a set of slave problems that are much easier to handle, both because of their much simpler energy functions and because of the absence of latent variables. To perform this training, we will apply the iterative projected subgradient method that updates $\{\mathbf{w},\{\boldsymbol{\lambda}^k\}\}$ at each iteration as follows

$$\mathbf{w} \leftarrow \mathbf{w} - s_t\delta_{\mathbf{w}} \ , \ \boldsymbol{\lambda}^k \leftarrow \text{proj}_{\boldsymbol{\Lambda}^k}(\boldsymbol{\lambda}^k - s_t\delta_{\boldsymbol{\lambda}^k}) \ , \tag{30}$$

where $\{\delta_{\mathbf{w}},\{\delta_{\boldsymbol{\lambda}^k}\}\}$ denotes a subgradient of function (28) at $\{\mathbf{w},\{\boldsymbol{\lambda}^k\}\}$, and $\text{proj}_{\boldsymbol{\Lambda}^k}(\cdot)$ denotes projection onto $\boldsymbol{\Lambda}^k$.

As described in the next lemma (see [1] for a proof), the above update essentially requires to compute an optimal solution for the slave subproblems, and, as we shall see, this computation can be performed very efficiently thanks to the much simpler energy functions involved (note that this would not be possible had we to deal with the energy of the master problem $\bar{E}^k$).

Lemma 1 *Let $\hat{\mathbf{x}}^{k,p}$, $\hat{\mathbf{x}}^{k,C}$ be binary minimizers of the energy functions $\bar{E}^k_p$, $\bar{E}^k_C$. Define $f^k_{pq} \equiv f_{pq}(\mathbf{y}^k)$, $\hat{X}^k_q \equiv \hat{x}^{k,C}_{qq} + \sum_p \hat{x}^{k,p}_{qq}, \forall q \in C$. Update (30) then reduces to*

$$\begin{bmatrix} \mathbf{w} \\ \lambda^k_{pq} \\ \lambda^k_{Cq} \end{bmatrix} -= s_t \begin{bmatrix} \tau\nabla J(w) + \sum_k \delta^k_{\mathbf{w}} \\ \frac{\hat{X}^k_q}{|S^k|+1} - \hat{x}^{k,p}_{qq} \\ \frac{\hat{X}^k_q}{|S^k|+1} - \hat{x}^{k,C}_{qq} \end{bmatrix}, \tag{31}$$

where $\delta^k_{\mathbf{w}} = \sum_{p,q} x^k_{pq} f^k_{pq} - \sum_{p\neq q} \hat{x}^{k,p}_{pq} f^k_{pq} - \frac{\sum_q \hat{X}^k_q f^k_{qq}}{|S^k|+1}$.

4.2.1 Solving the slave problems

Despite the fact that the potentials included in the energy functions of the slave problems are of high-arity (*e.g.*, $\bar{\phi}_p(\cdot)$, $\bar{\phi}_C(\cdot)$ are functions of $|S^k|$ and $|C|$ variables respectively), the special structure of these potentials allows us to compute a binary minimizer for the energies $\bar{E}^k_p$, $\bar{E}^k_C$ very efficiently. This is detailed in the following lemma (proved in [1]).

Lemma 2 *Let $[a]_+ \equiv \max(a,0)$, $[a]_- \equiv \min(a,0)$.*

1. *For fixed p, let $\theta^k_q \equiv \frac{\bar{u}^k_{qq}(1)}{|S^k|+1} + \lambda^k_{pq}, \forall q$ and let us define $\bar{\theta}^k_q \equiv \bar{u}^k_{pq}(1) + [\theta^k_q]_+, \forall q \neq p$ and $\bar{\theta}^k_p = \theta^k_p$. A minimizer $\hat{\mathbf{x}}$ of $\bar{E}^k_p(\mathbf{x};\mathbf{w},\boldsymbol{\lambda}^k)$ can be computed as follows:*

$$\forall q \neq p, \ \hat{x}_{qq} \leftarrow [\theta^k_q < 0] \tag{32}$$

$$\forall q, \ \hat{x}_{pq} \leftarrow [q = \bar{q}], \ \textit{where } \bar{q} = \arg\min_q \bar{\theta}^k_q \tag{33}$$

Data: training samples $\{\mathbf{y}^k, C^k, S^k\}_{k=1}^K$, features $\{f_{pq}(\cdot)\}$

$\boldsymbol{\lambda}^k \leftarrow \mathbf{0},\ \forall k$
repeat
  /* Optimize over $\mathbf{x}^k$ */
  compute optimal set of exemplars Q^k via (29)
  set $x^k_{qq}=1 \Leftrightarrow q \in Q^k,\ \ x^k_{pq}=1 \Leftrightarrow q=\arg\min_{q\in Q^k} d^k_{p,q},\ \forall p \neq q$
  /* Apply T rounds of projected subgradient */
  repeat T times {
    get solutions $\hat{\mathbf{x}}^{k,p}$, $\hat{\mathbf{x}}^{k,C}$ of slaves $\bar{E}^k_p$, $\bar{E}^k_C$ via (32)-(34)
    update $\mathbf{w}, \boldsymbol{\lambda}^k$ via (31)
  }
until convergence

Fig. 1: Pseudocode for the clustering learning algorithm.

2. *For fixed* $C \in \mathcal{C}^k$, *let* $\theta^k_q \equiv \frac{\bar{u}^k_{qq}(1)}{|S^k|+1} + \lambda^k_{Cq}, \forall q \in C$. *A minimizer* $\hat{\mathbf{x}}$ *of* $\bar{E}^k_C(\mathbf{x}; \mathbf{w}, \boldsymbol{\lambda}^k)$ *is given by*

$$\forall q \in C,\ \hat{x}_{qq} = \begin{cases} [\theta^k_q < \alpha], & \text{if } 2\alpha + \sum_{q' \in C} [\theta^k_{q'} - \alpha]_- < 0 \\ 0, & \text{otherwise} \end{cases} \tag{34}$$

The pseudocode of the resulting learning algorithm for clustering appears in Fig. 1. As can be seen, this algorithm alternates between filling in the latent variables $\mathbf{x}^k$ and updating $\mathbf{w}$, $\boldsymbol{\lambda}$ via doing a parallel training for the easy-to-handle slave problems. For the latter task, T rounds of the projected subgradient algorithm are being used. Note that in practice the number of rounds T can be set to be very small (*e.g.*, even 1). In addition, if we want to further improve the efficiency of the above method, we can make use of a stochastic subgradient algorithm. In this case the only difference is that at each iteration we must randomly pick just one training sample, say the k-th one, and then update only $\boldsymbol{\lambda}^k, \mathbf{x}^k, \mathbf{w}$ during the current iteration (we must also replace the term $\sum_k \delta^k_{\mathbf{w}}$ in (31) with just $\delta^k_{\mathbf{w}}$). Overall, this leads to an extremely efficient learning scheme for clustering.

5. Training high-order latent CRFs

Although the focus of this work is on clustering, we note that our learning method can be generalized to an extremely broad class of problems that play an important role in many vision applications: namely *high-order CRFs with latent variables* (it thus also extends the recently proposed learning framework in [6]). In the following we briefly explain how our method handles such latent models. Let E_{G^k} denote the energy of a CRF defined on a hypergraph $G^k = (V^k, \mathcal{E}^k)$ with vertices V^k and hyperedges $\mathcal{E}^k$, where

$$E_{G^k}(\mathbf{x}, \mathbf{z}; \mathbf{w}) = \sum_{p \in V^k} u^k_p(x_p, z_p; \mathbf{w}) + \sum_{e \in \mathcal{E}^k} \phi^k_e(\mathbf{x}_e, \mathbf{z}_e; \mathbf{w}).$$

Functions $u^k_p(\cdot)$, $\phi^k_e(\cdot)$ denote the unary and higher-order potentials that are expressible in terms of an unknown vector of parameters $\mathbf{w}$ and some feature functions of the input data $\mathbf{y}^k$ (*e.g.*, $u^k_p(x_p, z_p; \mathbf{w}) = \mathbf{w}^T f_p(x_p, z_p, \mathbf{y}^k)$ and similarly for $\phi^k_e(\cdot)$). We want to estimate $\mathbf{w}$ based on a set of training samples $\{\mathbf{y}^k, \mathbf{z}^k\}_{k=1}^K$, where only variables $\mathbf{z}^k$ are observed during training (*i.e.*, variables $\mathbf{x}^k$ are hidden).

The only requirement that must hold in this case is that an error function $\Delta(\mathbf{x}, \mathbf{z}; \mathbf{z}^k)$ must be used that fulfills the following property: the function $E_{G^k}(\mathbf{x}, \mathbf{z}; \mathbf{w}) - \Delta(\mathbf{x}, \mathbf{z}; \mathbf{z}^k)$ should equal the energy of a high-order CRF defined on a hypergraph $\bar{G}^k = (\bar{V}^k, \bar{\mathcal{E}}^k)$, *i.e.*, it must hold $E_{G^k}(\mathbf{x}, \mathbf{z}; \mathbf{w}) - \Delta(\mathbf{x}, \mathbf{z}; \mathbf{z}^k) = \bar{E}_{\bar{G}^k}(\mathbf{x}, \mathbf{z}; \mathbf{w})$, where

$$\bar{E}_{\bar{G}^k}(\mathbf{x}, \mathbf{z}; \mathbf{w}) = \sum_{p \in \bar{V}^k} \bar{u}^k_p(x_p, z_p; \mathbf{w}) + \sum_{e \in \bar{\mathcal{E}}^k} \bar{\phi}^k_e(\mathbf{x}_e, \mathbf{z}_e; \mathbf{w}).$$

Note that this is an extremely loose assumption given that both the hypergraph $\bar{G}^k$ and the new potentials $\bar{u}^k_p(\cdot)$, $\bar{\phi}^k_e(\cdot)$ may be totally different than G^k or $u^k_p(\cdot)$, $\phi^k_e(\cdot)$ respectively.

In such a case we can proceed by applying the dual decomposition method to the CRF energy $\bar{E}_{\bar{G}^k}$ [7]. In its simplest version this entails choosing an arbitrary partition $\{\bar{G}^k_i = (\bar{V}^k_i, \bar{\mathcal{E}}^k_i)\}$ of $\bar{G}^k$ and defining on each $\bar{G}^k_i$ a slave CRF with energy $\bar{E}_{\bar{G}^k_i}(\mathbf{x}, \mathbf{z}; \mathbf{w}, \boldsymbol{\lambda}^{k,i})$ whose potentials $\bar{u}^{k,i}_p(x_p, z_p; \mathbf{w}, \boldsymbol{\lambda}^{k,i})$, $\bar{\phi}^{k,i}_e(\mathbf{x}_e, \mathbf{z}_e; \mathbf{w})$ are given by

$$\bar{u}^{k,i}_p(\cdot; \mathbf{w}, \boldsymbol{\lambda}^{k,i}) = \lambda^{k,i}_p(\cdot) + \frac{\bar{u}^k_p(\cdot; \mathbf{w})}{|\{i : p \in \bar{V}^k_i\}|},\ \ \bar{\phi}^{k,i}_e = \bar{\phi}^k_e.$$

Here, the dual variables $\boldsymbol{\lambda}^k = \{\boldsymbol{\lambda}^{k,i}\}$ are assumed to belong to the set $\boldsymbol{\Lambda}^k = \{\boldsymbol{\lambda}^k : \sum_{i: p \in \bar{V}^k_i} \lambda^{k,i}_p(\cdot) = 0,\ \forall p\}$.

By then leveraging the convex dual relaxation resulting from this decomposition, we can show that the original intractable loss, which in this case equals $\min_{\mathbf{x}^k, \mathbf{w}} \tau J(\mathbf{w}) + \sum_k \bar{\mathcal{L}}_{\bar{E}_{\bar{G}^k}}(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w})$, can be approximated with the following much easier-to-handle tight upper bound[2]

$$\min_{\mathbf{x}^k, \mathbf{w}, \boldsymbol{\lambda}^k \in \boldsymbol{\Lambda}^k} \tau J(\mathbf{w}) + \sum_{k,i} \bar{\mathcal{L}}_{\bar{E}_{\bar{G}^k_i}}(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}, \boldsymbol{\lambda}^{k,i}) \tag{35}$$

that relates to the training of CRF slaves $\bar{E}_{\bar{G}^k_i}(\,\cdot\,; \mathbf{w}, \boldsymbol{\lambda}^{k,i})$.

In this manner a very efficient learning scheme can be derived that alternates between the following two steps: (a) filling in the latent variables by setting $\mathbf{x}^k = \arg\min_{\mathbf{x}} \bar{E}_{\bar{G}^k}(\mathbf{x}, \mathbf{z}^k; \mathbf{w})$ (or more generally by simply updating $\mathbf{x}^k$ such that the energy $\bar{E}_{\bar{G}^k}(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w})$ decreases), and (b) updating $\mathbf{w}$, $\boldsymbol{\lambda}^k$ by training in parallel the CRF slaves $\bar{E}_{\bar{G}^k_i}$ via T rounds of projected subgradient, where each round essentially reduces to computing an optimal solution for each slave $\bar{E}_{\bar{G}^k_i}$ (this comprises the main step for the subgradient computation).

Besides its extreme efficiency, the above learning scheme also provides great flexibility and generality, which

[2] $\bar{\mathcal{L}}_{\bar{E}}(\mathbf{x}', \mathbf{z}') \equiv \bar{E}(\mathbf{x}', \mathbf{z}') - \min_{\mathbf{x}, \mathbf{z}} \bar{E}(\mathbf{x}, \mathbf{z})$ for any energy $\bar{E}(\mathbf{x}, \mathbf{z})$.

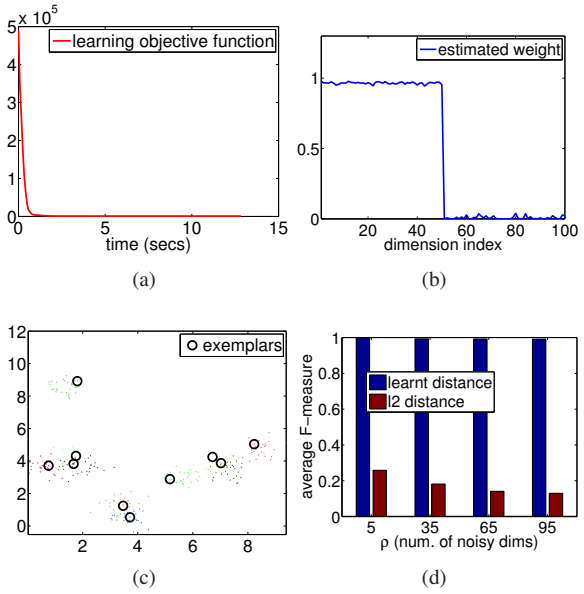


Fig. 2: In (c) we show a projection of a test dataset onto 2 non-noisy dimensions. The test set lives in $\mathbb{R}^{100}$ and consists of 10 ground truth clusters indicated with different colors. Circle centers denote the estimated exemplars when using the learnt weights shown in (b). (See also text).

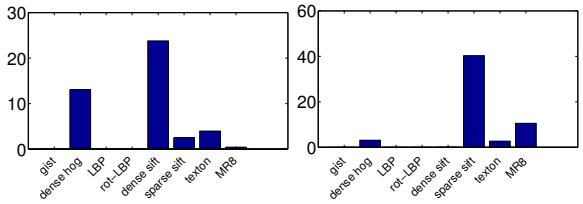
stems from the freedom of choice with regard to the slave CRFs that are used in each case. For instance, even if the trianing of the original latent CRF is quite complex, we can decompose it to the much easier training of simpler slaves (*e.g.*, one slave per clique). In this manner we are able to efficiently handle the training of a very broad class of CRFs with latent variables. Furthermore, through a proper selection of the slaves, the above framework can also be easily adapted to take advantage of the special structure that may exist in classes of latent CRFs used in practice, *e.g.*, submodular CRFs (note that essentially the only requirement is that one can compute optimal solutions for the slave CRFs).

6. Experimental results

To verify the effectiveness of our learning framework for clustering, we conducted various experiments. In all of them we assume that the penalties $\{d_{qq}\}$ are set to a fixed value and so only the distances $\{d_{pq}\}_{p\neq q}$ depend on $\mathbf{w}$. We also used an l_1-norm regularizer $R(\mathbf{w})$ - as we found it to lead to better generalization performance compared to a quadratic regularizer (probably due to sparsity) - and set $\alpha = \beta = 1$ in (15). For obtaining a clustering at test time (*i.e.*, after we learn $\mathbf{w}$), any of the algorithms capable of optimizing objective function (1) can be used (*e.g.*, [9]).

Our first experiment is on clustering synthetic data. In this case each synthetic dataset was generated by sampling N data points (living in $\mathbb{R}^D$) from a mixture of M multi-dimensional Gaussian distributions with a diagonal covariance matrix $\Sigma = \mathrm{diag}(\sigma_i)$. The values σ_i were chosen such that a percentage ρ of the total D dimensions were noisy (*i.e.*, their variance was set to be much larger compared to the rest of the dimensions). Clustering such a set of datapoints requires learning a distance function of the following form $d_{pq} = \sum_{i=1}^{D} w_i (x_p^i - x_q^i)^2$, where $\mathbf{w} = \{w_i\}$ is the vector of unknown weights. To this end, a set of ground truth partitions of sampled data is assumed to be given as input. Based on the above setting, we performed multiple tests by varying the number of total dimensions D, the percentage of corrupted dimensions ρ, as well as the number of clusters M and the number of datapoints N (each time 10 ground truth partitions of different datasets were used as training data, and 10 test datasets were clustered based on the estimated $\mathbf{w}$). Fig. 2(a) shows a typical example of how the learning objective function varies in this case when running our algorithm (we show an average over 20 different runs). Notice the extremely fast convergence of our method. The number of dimensions D used in this example was $D = 100$, the last half of which were corrupted (*i.e.*, $\rho = 50\%$), while the other first half had equal variance (N and M varied per run). The resulting vector $\mathbf{w} = \{w_i\}$ estimated by our method is shown in Fig. 2(b), where the weights corresponding to the noisy dimensions are the last ones. Notice how the algorithm has successfully managed to significantly suppress the values of all these weights (we note that the use of a sparsity inducing regularizer $R(\mathbf{w}) = ||\mathbf{w}||_1$ was important to achieve this). We also show in Fig. 2(c) a typical clustering result produced for a test dataset using the learnt distance. As can be seen, the clustering algorithm has also managed to automatically determine the correct number of exemplars (in this case 500 datapoints had been sampled from a mixture of 10 Gaussians, *i.e.*, $M = 10$, $N = 500$). The above successful behavior of our learning method was observed throughout all our tests. To this end, we show in Fig. 2(d) the average accuracy (F-measure) obtained for clusterings of test datasets for different values of ρ (we note that the same ρ has been used in both training and testing and the average is over 10 runs). As can be observed, even for very large ρ our method manages to correctly estimate $\mathbf{w}$ and thus learn distances that yield highly accurate clusterings during testing (*i.e.*, F-measure ≈ 1). On the contrary, squared Euclidean distance fails completely in this case, as expected.

The next experiment is about learning a distance function for texture classification, *i.e.*, for clustering different classes of texture images. In this case the distance $d(\cdot)$ to be estimated equals a weighted combination of known distances, *i.e.*, it holds $d(\cdot) = \sum_f w_f d^f(\cdot)$, where each individual distance $d^f(\cdot)$ compares images based on a different visual feature f. Due to the generality of our learning framework, arbitrary distances $d^f(\cdot)$ can be used (*e.g.*, non-metric, non-differentiable or even asymmetric distance functions can be



(a) Outex (b) UIUC

Fig. 3: Learnt weights for Outex and UIUC.

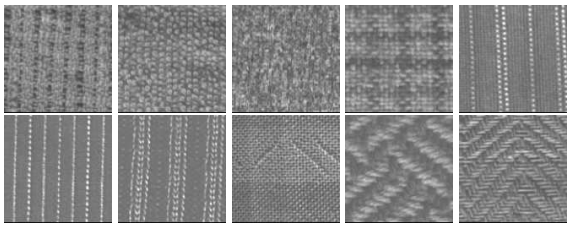


Fig. 4: 10 of 24 estimated exemplar images for Outex.

chosen). Furthermore, due to the robustness of our learning procedure, we can freely use as many feature distance functions as we want and let the learning algorithm figure out the ones that are useful in clustering. We first apply our method to images from the Outex database [13] that contains 24 different texture classes. We use 10 train/test splits, where each time a single partition containing half of the images from each class is used as training data while the rest of the images are clustered during testing. The visual features that are used are the following: gist, histogram of dense quantized HOGs, local binary patterns (LBP), rotation invariant LBP (rot-LBP) [2], histograms of quantized dense SIFT descriptors, histograms of quantized sparse SIFT descriptors (at Hessian-affine interest points), standard texton histograms and histograms based on MR8 textons [16]. For all of them we use the chi-squared distance (χ^2), except for gist (squared euclidean distance) and rot-LBP (l_1 distance). The weights learnt by our algorithm appear in Fig. 3(a). Based on these weights, clustering attains an accuracy of 100% during testing (fig. 4 shows 10 of the 24 images that were automatically determined as exemplars for one of the test sets). In this case, due to the fact that texture classes exhibit no rotational variations, rotation invariant features such as rot-LBP or MR8 are suppressed by our method and essentially dense HOG and dense SIFT features suffice for comparing texture images.

On the contrary, if we apply our learning algorithm to the UIUC texture database [11] (consisting of 25 texture classes) the resulting set of learnt weights appears in Fig. 3(b). Essentially, due to the existence of intra-class rotational variations, dense HOG and SIFT features (which offer no rotational invariance) have been traded off for the rotationally invariant sparse SIFT and MR8 features. In this case, the average clustering accuracy of our method is 86%. Furthermore, nearest neighbor classification based on the

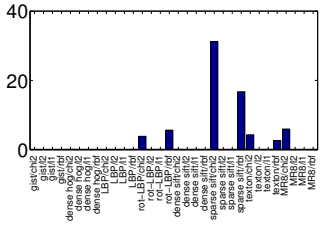
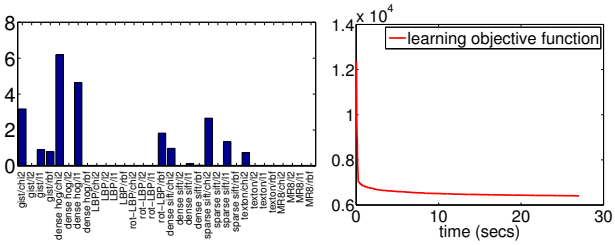


Fig. 5: Learnt weights for multiple distance-feature pairs in UIUC



(a) (b)

Fig. 6: **(a)** Learnt weights for multiple distance-feature pairs for the scene dataset. **(b)** Evolution of the learning objective function for the scene dataset.

learnt weighted distance achieves an accuracy of 95.3%, whereas the best obtained accuracy using either a single feature distance or the average of all feature distances does not even exceed 90%. In fact, the difference in performance becomes even greater when there exist additional redundant or noisy features. To further explore the benefit of using our learning framework in such a case, we performed the following experiment: in addition to the above mentioned features, we artificially added an equal number of noisy features by randomly sampling vectors from a Gaussian distribution. What we observed is that our method had no problem at all to completely suppress all the weights of irrelevant features, thus achieving the same level of accuracy as before. On the contrary, in this case performance dropped significantly when the average of all feature distances was used for classifying images.

In our next experiment we went one step further by allowing not only different visual features but also multiple distance functions per feature (apparently even important features can be misused if they are not compared properly). In this case the distance function for clustering is given by $d(\cdot) = \sum_{f,i} w_{f,i} d^{f,i}(\cdot)$, where i now runs over multiple distances per feature f. To this end, we used the following distances for all of the features: χ^2 distance (denoted chi2 in the figures), squared Euclidean distance (denoted l_2), l_1-norm distance, and the RBF kernel distance[3]. Using this setting and applying again our learning method to the UIUC dataset leads to the new weights in Fig. 5. Essentially, the main difference is that the rot-LBP feature, which was originally suppressed due to the use of a l_1 distance, is

[3]We define the distance of a similarity kernel $k(\cdot,\cdot)$ as $d^k(\mathbf{x},\mathbf{y}) = \sqrt{k(\mathbf{x},\mathbf{x}) + k(\mathbf{y},\mathbf{y}) - k(\mathbf{x},\mathbf{y}) - k(\mathbf{y},\mathbf{x})}$.

Fig. 7: 10 of the exemplar images estimated for the Scene dataset.

now given more weight when used in conjunction with, *e.g.*, the RBF kernel distance. This also has the effect of raising the accuracy of nearest neighbor classification to 96.1%.

Using this idea of multiple distances per feature, we also applied our method to the 15 scene category dataset for scene classification [12]. To this end, we chose 10 random train/test splits containing 80 images per scene (with half of the images being in the training set). The resulting evolution of the objective function during learning (averaged over the 10 runs) appears in Fig. 6(b). It can be noticed that convergence is again very fast. Also, the estimated weights for various feature-distance combinations are shown in Fig. 6(a). As expected, due to the greater intra-class image variation, a larger number of features come into play compared to texture classification. In this case, the average clustering accuracy is 63%. Furthermore, using the resulting weighted distance in conjunction with a nearest neighbor classifier yields a NN accuracy of 70.2%. The important thing to note here is that this provides a significant improvement compared to the best NN accuracy that can be obtained using either a single feature-distance combination or the average distance, which in this case does not go over 61%. Fig. 7 also shows a few of the exemplar images that were chosen when clustering the test images for one of the splits of the Scene dataset.

We also experimentally compared our method with [4], which is a state of the art learning algorithm for clustering (our implementation of [4] was based on adapting the public SVM-struct software). In all experiments we found our method to have consistently better performance. For instance, the clustering accuracy of [4] for the UIUC and Scene experiments were 74% and 52% respectively (vs 86% and 63% for our method). Similarly, the corresponding NN accuracy of [4] were 82.1% and 62.1% (vs 95.3% and 70.2% for our method). We also compared our method with the state of the art multiple kernel learning algorithm in [15]. The classification accuracy of that algorithm for the same UIUC and Scene experiments were 94.9% and 63.4% respectively (vs 95.3% and 70.2% for our method).

We should add that in all the experiments we imposed a positivity constraint on $\mathbf{w}$, *i.e.*, we required $\mathbf{w} \geq 0$. Note that our method can naturally handle any constraint of the form $\mathbf{w} \in W$ with W a convex set. To this end, one must simply replace the update $\mathbf{w} \leftarrow \mathbf{w} - s_t \delta_{\mathbf{w}}$ in (30) with the update $\mathbf{w} \leftarrow \text{proj}_W(\mathbf{w} - s_t \delta_{\mathbf{w}})$, where $\text{proj}_W(\cdot)$ denotes projection onto set W (in the case where $W = \{\mathbf{w} : \mathbf{w} \geq 0\}$, it holds $\text{proj}_W(\mathbf{w}) = \max(\mathbf{w}, 0)$).

7. Conclusions

In this paper we presented a discriminative learning framework for distance-based clustering. The proposed framework is very general, highly efficient (as it relies on a very fast subgradient scheme) and parallelizable. It allows learning a very broad class of distances that may depend on a large number of unknown parameters. Furthermore, we showed that it can be extended to efficiently handle the training of a very broad class of high-order models that play an important role in computer vision: these are high-order CRFs with latent variables. Due to the widespread use of both distance-based clustering and latent CRFs, we firmly believe that our framework will be a valuable tool for many vision applications[4].

References

[1] Supplemental material. 4 (http://www.csd.uoc.gr/~komod/publications/ docs/iccv_2011.pdf).

[2] T. Ahonen, J. Matas, C. He, and M. Pietikäinen. Rotation invariant image description with local binary pattern histogram fourier features. In *SCIA*, 2009. 7

[3] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 2009. 1

[4] T. Finley and T. Joachims. Supervised clustering with support vector machines. In *ICML*, 2005. 1, 8

[5] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 2007. 2

[6] N. Komodakis. Efficient training for pairwise or higher order CRFs via dual decomposition. In *CVPR*, 2011. 1, 5

[7] N. Komodakis and N. Paragios. Beyond pairwise energies: Efficient optimization for higher-order MRFs. In *CVPR*, 2009. 3, 5

[8] N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *ICCV*, 2007. 1

[9] N. Komodakis, N. Paragios, and G. Tziritas. Clustering via LP-based Stabilities. In *NIPS*, 2008. 2, 6

[10] N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *PAMI*, 2010. 1, 3

[11] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *PAMI*, 2005. 7

[12] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 8

[13] T. Ojala, T. Mäenpää, M. Pietikäinen, J. Viertola, J. Kyllönen, and S. Huovinen. Outex - new framework for empirical evaluation of texture analysis algorithms. In *ICPR*, 2002. 7

[14] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *NIPS*, 2004. 2

[15] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *ICCV*, 2007. 8

[16] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *IJCV*, 2005. 7

[17] C.-N. J. Yu and T. Joachims. Learning structural svms with latent variables. In *ICML*, 2009. 1

[4] An implementation of our clustering learning algorithm will be available from http://www.csd.uoc.gr/~komod