Network Flow Algorithms for Structured Sparsity

Authored by:

Francis R. Bach
Julien Mairal
Guillaume R. Obozinski
Rodolphe Jenatton

Abstract

We consider a class of learning problems that involve a structured sparsity-inducing norm defined as the sum of ell_infty-norms over groups of variables. Whereas a lot of effort has been put in developing fast optimization methods when the groups are disjoint or embedded in a specific hierarchical structure, we address here the case of general overlapping groups. To this end, we show that the corresponding optimization problem is related to network flow optimization. More precisely, the proximal problem associated with the norm we consider is dual to a quadratic min-cost flow problem. We propose an efficient procedure which computes its solution exactly in polynomial time. Our algorithm scales up to millions of groups and variables, and opens up a whole new range of applications for structured sparse models. We present several experiments on image and video data, demonstrating the applicability and scalability of our approach for various problems.

1 Paper Body

Sparse linear models have become a popular framework for dealing with various unsupervised and supervised tasks in machine learning and signal processing. In such models, linear combinations of small sets of variables are selected to describe the data. Regularization by the ?1 -norm has emerged as a powerful tool for addressing this combinatorial variable selection problem, relying on both a well-developed theory (see [1] and references therein) and efficient algorithms [2, 3, 4]. The ?1 -norm primarily encourages sparse solutions, regardless of the potential structural relationships (e.g., spatial, temporal or hierarchical) existing between the variables. Much effort has recently been devoted to designing sparsity-inducing regularizations capable of encoding higher-order information about allowed patterns of non-zero coefficients [5, 6, 7, 8, 9, 10], with successful applications in bioinformatics [6, 11], topic modeling [12] and computer vision [9, 10]. By considering sums of norms of appropriate subsets, or groups, of variables, these regularizations control the sparsity patterns of the solutions. The underlying optimization problem is usually difficult, in part because it involves nonsmooth components. Proximal methods have proven to be effective in this context, essentially because of their fast convergence rates and their scalability [3, 4]. While the settings where the penalized groups of variables do not overlap or are embedded in a treeshaped hierarchy [12] have already been studied, regularizations with general overlapping groups have, to the best of our knowledge, never been addressed with proximal methods. This paper makes the following contributions: ? It shows that the proximal operator associated with the structured norm we consider can be ? ?

Contributed equally. Laboratoire d?Informatique de l?Ecole Normale Sup?erieure (INRIA/ENS/CNRS UMR 8548)

1

computed with a fast and scalable procedure by solving a quadratic min-cost flow problem. ? It shows that the dual norm of the sparsity-inducing norm we consider can also be evaluated efficiently, which enables us to compute duality gaps for the corresponding optimization problems. ? It demonstrates that our method is relevant for various applications, from video background subtraction to estimation of hierarchical structures for dictionary learning of natural image patches.

2

Structured Sparse Models

We consider in this paper convex optimization problems of the form min f (w) + ??(w),

w?Rp

(1)

where f : Rp ? R is a convex differentiable function and ? : Rp ? R is a convex, nonsmooth, sparsity-inducing regularization function. When one knows a priori that the solutions of this learning problem have only a few non-zero coefficients, ? is often chosen to be the ?1 -norm (see [1, 2]). When these coefficients are organized in groups, a penalty encoding explicitly this prior knowledge can improve the prediction performance and/or interpretability of the learned models [13, 14]. Denoting by G a set of groups of indices, such a penalty might for example take the form: X X ?(w) , ?g max —wj — = ?g kwg k? , (2) g?G

j?g

g?G

where wj is the j-th entry of w for j in [1; p] , {1, . . . , p}, the vector wg in R—g— records the coefficients of w indexed by g in G, and the scalars ?g are positive weights. A sum of ?2 -norms is also used in the literature [7], but the ?? -norm is piecewise linear, a property that we take advantage of in this paper. Note that when G is the set of singletons of [1; p], we get back the ?1 -norm. If G is a more general partition of [1; p], variables are selected in groups rather than individually. When the groups overlap, ? is still a norm and sets groups of variables to zero together [5]. The latter setting has first been considered for hierarchies [7, 11, 15], and then extended to general group structures [5].1 Solving Eq. (1) in this context becomes challenging and is the topic of this paper. Following Jenatton et al. [12] who tackled the case of hierarchical groups, we propose to approach this problem with proximal methods, which we now introduce. 2.1

Proximal Methods

In a nutshell, proximal methods can be seen as a natural extension of gradient-based techniques, and they are well suited to minimizing the sum f + ?? of two convex terms, a smooth function f ?continuously differentiable with Lipschitz-continuous gradient? and a potentially non-smooth function ?? (see [16] and references therein). At each iteration, the function f is linearized at the current estimate w0 and the so-called proximal problem has to be solved: L min f (w0) + (w ? w0)? ?f (w0) + ??(w) + kw ? w0 k22 . w?Rp 2 The quadratic term keeps the solution in a neighborhood where the current linear approximation holds, and L ¿ 0 is an upper bound on the Lipschitz constant of ?f . This problem can be rewritten as 1 2 min ku ? wk2 + ?? ?(w), (3) w?Rp 2 with ?? , ?/L, and u , w0 ? L1 ?f (w0). We call proximal operator associated with the regularization ?? ? the function that maps a vector u in Rp onto the (unique, by strong convexity) solution w? of Eq. (3). Simple proximal methods use w? as the next iterate, but accelerated variants [3, 4] are also based on the proximal operator and require to solve problem (3) exactly and efficiently to enjoy their fast convergence rates. Note that when ? is the ?1 -norm, the solution of Eq. (3) is obtained by soft-thresholding [16]. The approach we develop in the rest of this paper extends [12] to the case of general overlapping groups when ? is a weighted sum of ?? -norms, broadening the application of these regularizations to a wider spectrum of problems.2 1 Note that other types of structured sparse models have also been introduced, either through a different norm [6], or through non-convex criteria [8, 9, 10]. 2 For hierarchies, the approach of [12] applies also to the case of where ? is a weighted sum of ?2 -norms.

2

3

A Quadratic Min-Cost Flow Formulation

In this section, we show that a convex dual of problem (3) for general overlapping groups G can be reformulated as a quadratic min-cost flow problem. We present an efficient algorithm to solve it exactly, as well as a related algorithm to compute the dual norm of ?. We start by considering the dual formulation to problem (3) introduced in [12], for the case where ? is a sum of ?? -norms: Lemma 1 (Dual of the proximal problem [12]) Given u in Rp , consider the problem X 1

2 min u? ? g s.t. ?g ? G, k? g k1 ? ??g

2 ??Rp?—G— 2

and

? gj = 0 if j ? / g,

(4)

g?G

where ? = (? g)g?G is in Rp?—G— , and ? gj denotes the j-th coordinate of the vector ? g . Then, every P solution ? ? = (? ?g)g?G of Eq. (4) satisfies w? = u? g?G ? ?g , where w? is the solution of Eq. (3). Without loss of generality,3 we assume from now on that the scalars uj are all non-negative, and we constrain the entries of ? to be non-negative. We now introduce a graph modeling of problem (4). 3.1

Graph Model

Let G be a directed graph G = (V, E, s, t), where V is a set of vertices, E ? V ? V a set of arcs, s a source, and t a sink. Let c and c? be two functions on the arcs, c : E ? R and c? : E ? R+ , where c is a cost function and c? is a non-negative capacity function. A flow is a non-negative function on arcs that satisfies capacity constraints on all arcs (the value of the flow on an arc is less than or equal to the arc capacity) and conservation constraints on all vertices (the sum of incoming flows at a vertex is equal to the sum of outgoing flows) except for the source and the sink. We introduce a canonical graph G associated with our optimization problem, and uniquely characterized by the following construction: (i) V is the union of two sets of vertices Vu and Vgr , where Vu contains exactly one vertex for each index j in [1; p], and Vgr contains exactly one vertex for each group g in G. We thus have —V — = —G— + p. For simplicity, we identify groups and indices with the vertices of the graph. (ii) For every group g in G, E contains an arc (s, g). These arcs have capacity ??g and zero cost. (iii) For every group g in G, and every index j in g, E contains an arc (g, j) with zero cost and infinite capacity. We denote by ? gj the flow on this arc. (iv) For every index j in [1; p], E contains an arc (j, t) with infinite capacity and a cost cj , 21 (uj ? ??j)2 , where ??j is the flow on (j, t). Note that by flow conservation, we necesP sarily have ??j = g?G ? gj .

Examples of canonical graphs are given in Figures 1(a)-(c). The flows ? gj associated with G can now be identified with the variables of problem (4): indeed, the sum of the costs on the edges leading to the sink is equal to the objective function of (4), while the capacities of the arcs (s, g) match the constraints on each group. This shows that finding a flow minimizing the sum of the costs on such a graph is equivalent to solving problem (4). When some groups are included in others, the canonical graph can be simplified to yield a graph with a smaller number of edges. Specifically, if h and g are groups with h ? g, the edges (g, j) for j ? h carrying a flow ? gj can be removed and replaced by a single edge (g, h) of infinite capacity and P zero cost, carrying the flow j?h ? gj . This simplification is illustrated in Figure 1(d), with a graph ? equivalent to the one of Figure 1(c). This does not change the optimal value of ?? , which is the quantity of interest for computing the optimal primal variable w? (a proof and a formal definition of these equivalent graphs are available in a longer technical report [17]). These simplifications are useful in practice, since they reduce the number of edges in the graph and improve the speed of the algorithms we are now going to present. 3 Let ?? denote a solution of Eq. (4). Optimality conditions of Eq. (4) derived in [12] show that for all j in [1; p], the signs of the non-zero coefficients ??g j for g in G are the same as the signs of the entries uj . To solve Eq. (4), one can therefore flip the signs of the negative variables uj

, then solve the modified dual formulation (with non-negative variables), which gives the magnitude of the entries ??g j (the signs of these being known).

3

s

s

? g1 +? g2 +? g3 ? ??g

? g1 +? g2 ? ??g

g ? g1 u1

? g2

g

??2 , c2

? g1

? g3

u2 ??1 , c1

? h2 +? h3 ? ??h

u3

? g2

u1

??3 , c3

h ? h3

? h2 u2

??1 , c1

u3

??2 , c2

??3 , c3

t

t

(a) G = {g = {1, 2, 3}}.

(b) G = {g = {1, 2}, h = {2, 3}}.

s

s ? h2 +? h3 ? ??h

? g1 +? g2 +? g3 ? ??g g ? g1 u1

u2 ??1 , c1

??2 , c2

g

h g ? h2 ? 3

? g2

? h2 +? h3 ? ??h

? g1 +? g2 +? g3 ? ??g

? g1

? h3 u3

u1

??3 , c3

h

? g2 +? h2

? g3 +? h3

u2 ??1 , c1
t
? g2 +? g3
??2 , c2
u3 ??3 , c3
t
(c) G = {g = {1, 2, 3}, h = {2, 3}}.
(d) G = {g = {1} ? h, h = {2, 3}}.
Figure 1: Graph representation of simple proximal problems with different group structures G. The three indices 1, 2, 3 are represented as grey squares, and the groups g, h in G as red discs. The source is linked to every group g, h with respective maximum capacity ??g , ??h and zero cost. Each variable uj is linked to the sink t, with an infinite capacity, and with a cost cj , 12 (uj ? ??j)2 . All other arcs in the graph have zero cost and infinite capacity. They represent inclusion relationships in-between groups, and between groups and variables. The graphs (c) and (d) correspond to a special case of tree-structured hierarchy in the sense of [12]. Their min-cost flow problems are equivalent. 3.2
Computation of the Proximal Operator
Quadratic min-cost flow problems have been well studied in the operations research literature [18]. One of the simplest cases, where G contains a single group g (? is the ?? -norm) as in Figure 1(a), can be solved by an orthogonal projection on the ?1 -ball of radius ??g . It has been shown that such a projection can be done in O(p) operations [18, 19]. When the group structure is a tree as in Figure 1(d), the problem can be solved in O(pd) operations, where d is the depth of the tree [12, 18].4 The general case of overlapping groups is more difficult. Hochbaum and Hong have shown in [18] that quadratic min-cost flow problems can be reduced to a specific parametric max-flow problem, for which an efficient algorithm exists [20].5 While this generic approach could be used to solve Eq. (4), we propose to use Algorithm 1 that also exploits the fact that our graphs have non-zero costs only on edges leading to the sink. As shown in the technical report [17], it has a significantly better performance in practice. This algorithm clearly shares some similarities with existing approaches in network flow optimization such as the simplified version of [20] presented in [21] that uses a divide and conquer strategy. Moreover, we have discovered after that this paper was accepted for publication that an equivalent algorithm exists for minimizing convex functions over polymatroid 4
When restricted to the case where ? is a sum of ?? -norms, the approach of [12] is in fact similar to [18]. By definition, a parametric max-flow problem consists in solving, for every value of a parameter, a maxflow problem on a graph whose arc capacities depend on this parameter. 5
4
sets [22]. This equivalence, however, requires a non-trivial representation of structured sparsityinducing norms with submodular functions, as recently pointed out by [23]. Algorithm 1 Computation of the proximal operator for overlapping groups. 1: Inputs: u ? Rp , a set of groups G, positive weights (?g)g?G , and ? (regularization parameter). 2: Build the initial graph G0

= (V0 , E0 , s, t) as explained in Section 3.2. 3: Compute the optimal flow: ?? ? computeFlow(V0 , E0). 4: Return: w = u ? ?? (optimal solution of the proximal problem). Function computeFlow(V = Vu ? Vgr , E) P P P 1: Projection step: ? ? arg min? j?Vu 12 (uj ? ? j)2 s.t. g?Vgr ?g . j?Vu ? j ? ? 2: For all nodes j in Vu , set ? j to be the capacity of the arc (j, t). 3: Max-flow step: Update (??j)j?Vu by computing a max-flow on the graph (V, E, s, t). 4: if ? j ? Vu s.t. ??j 6= ? j then 5: Denote by (s, V +) and (V ? , t) the two disjoint subsets of (V, s, t) separated by the minimum (s, t)-cut of the graph, and remove the arcs between V + and V ? . Call E + and E ? the two remaining disjoint subsets of E corresponding to V + and V ? . 6: (??j)j?Vu+ ? computeFlow(V + , E +). 7: (??j)j?Vu? ? computeFlow(V ? , E ?). 8: end if 9: Return: (??j)j?Vu . TheP intuition behind this algorithm is the following: The first step looks for a candidate value for ??= g?G ? g by solving a relaxed version of problem Eq. (4), where the constraints k? g k1 ? ??g are P ? ? 1?? dropped and replaced by a single one k?k g?G ?g . The relaxed problem only depends on ? 2 and can be solved in linear time. By calling its solution ?, it provides a lower bound ku ? ?k2 /2 on the optimal cost. Then, the second step tries to find a feasible flow of the original problem (4) such that the resulting vector ?? matches ?, which is in fact a max-flow problem [24]. If ?? = ?, then the cost of the flow reaches the lower bound, and the flow is optimal. If ?? 6= ?, the lower bound is not achievable, and we construct a minimum (s, t)-cut of the graph [25] that defines two disjoints sets of nodes V + and V ? ; V + is the part of the graph that could potentially have received more flow from the source (the arcs between s and V + are not saturated), whereas all arcs linking s to V ? are saturated. At this point, it is possible to show that the value of the optimal min-cost flow on all arcs between V + and V ? is necessary zero. Thus, removing them yields an equivalent optimization problem, which can be decomposed into two independent problems of smaller sizes and solved recursively by the calls to computeFlow(V + , E +) and computeFlow(V ? , E ?). A formal proof of correctness of Algorithm 1 and further details are relegated to [17]. The approach of [18, 20] is guaranteed to have the same worst-case complexity as a single max-flow algorithm. However, we have experimentally observed a significant discrepancy between the worst case and empirical complexities for these flow problems, essentially because the empirical cost of each max-flow is significantly smaller than its theoretical cost. Despite the fact that the worst-case guarantee of our algorithm is weaker than their (up to a factor —V —), it is more adapted to the structure of our graphs and has proven to be much faster in our experiments (see technical report [17]). Some implementation details are crucial to the efficiency of the algorithm: ? Exploiting connected components: When there exists no arc between two subsets of V , it is possible to process them independently in order to solve the global min-cost flow problem. ? Efficient max-flow algorithm: We have implemented the ?push-relabel? algorithm of [24] for solving our max-flow problems, using classical heuristics that significantly speed it up in practice (see [24, 26]). This algorithm leverages the concept of pre-flow that relaxes the definition of flow and allows vertices to have a positive excess. It can be initialized with any valid pre-flow, enabling warm-restarts when the max-flow is called several times as in our algorithm. ? ImprovedP projection step: The first P line of the function Preplaced by P computeFlow can be ? ? arg min? j?Vu 12 (uj ? ? j)2 s.t. g?j ?g . The g?Vgr ?g and —? j — ? ? j?Vu ? j ? ? P idea is that the structure of the graph will not allow ??j to be greater than ? g?j ?g after the maxflow step. Adding these additional constraints leads to better performance when the graph is not well balanced. This modified projection step can still be computed in linear time [19]. 5

3.3

Computation of the Dual Norm

The dual norm ?? of ?, defined for any vector ? in Rp by ?? (?) , max?(z)?1 z? ?, is a key quantity to study sparsity-inducing regularizations [5, 15, 27]. We use it here to monitor the convergence of the proximal method through a duality gap, and define a proper optimality criterion for problem (1). We denote by f ? the Fenchel conjugate of f [28], defined by f ? (?) , supz [z? ? ? f (z)]. The duality gap for problem (1) can be derived from standard Fenchel duality arguments [28] and it is equal to f (w) + ??(w) + f ? (??) for w, ? in Rp with ?? (?) ? ?. Therefore, evaluating the duality gap requires to compute efficiently ?? in order to find a feasible dual variable ?. This is equivalent to solving another network flow problem, based on the following variational formulation: X ?? (?) = min ? s.t. ? g = ?, and ?g ? G, k? g k1 ? ? ?g with ? gj = 0 if j ? / g. (5) ??Rp?—G—

g?G

In the network problem associated with (5), the capacities on the arcs (s, g), g ? G, are set to ? ?g , and the capacities on the arcs (j, t), j in [1; p], are fixed to ?j . Solving problem (5) amounts to finding the smallest value of ? , such that there exists a flow saturating the capacities ?j on the arcs leading to the sink t (i.e., ?? = ?). The algorithm below is proven to be correct in [17]. Algorithm 2 Computation of the dual norm. 1: Inputs: ? ? Rp , a set of groups G, positive weights (?g)g?G . 2: Build the initial graph G0 = (V0 , E0 , s, t) as explained in Section 3.3. 3: ? ? dualNorm(V0 , E0). 4: Return: ? (value of the dual norm). Function dualNorm(V = Vu ? Vgr , E) P P 1: ? ? (j?Vu ?j)/(g?Vgr ?g) and set the capacities of arcs (s, g) to ? ?g for all g in Vgr . 2: Max-flow step: Update (??j)j?Vu by computing a max-flow on the graph (V, E, s, t). 3: if ? j ? Vu s.t. ??j 6= ?j then 4: Define (V + , E +) and (V ? , E ?) as in Algorithm 1, and set ? ? dualNorm(V ? , E ?). 5: end if 6: Return: ? .

4

Applications and Experiments

Our experiments use the algorithm of [4] based on our proximal operator, with weights ?g set to 1. 4.1

Speed Comparison

We compare our method (ProxFlow) and two generic optimization techniques, namely a subgradient descent (SG) and an interior point method,6 on a regularized linear regression problem. Both SG and ProxFlow are implemented in C++. Experiments are run on a single-core 2.8 GHz CPU. We consider a design matrix X in Rn?p built from overcomplete dictionaries of discrete cosine transforms (DCT), which are naturally organized on one- or two-dimensional grids and display local correlations. The following families of groups G using this spatial information are thus considered: (1) every contiguous sequence of length 3 for the one-dimensional case, and (2) every 3?3-square in the two-dimensional setting. We generate vectors y in Rn according to the linear model y = Xw0 + ?, where ? ? N (0, 0.01kXw0 k22). The vector w0 has about 20% percent nonzero components, randomly selected, while respecting the structure of G, and uniformly generated between [?1, 1]. In our experiments, the regularization parameter ? is chosen to achieve the same sparsity as w0 . For SG, we take the step size to be equal to a/(k + b), where k is the iteration number, and (a, b) are the best parameters selected in {10?3 , . . . , 10}?{102 , 103 , 104 }. For the interior point methods, since problem (1) can be cast either as a quadratic (QP) or as a conic program (CP), we show in Figure 2 the results for both formulations. Our approach compares favorably with the other methods, on three problems of different sizes, (n, p) ? {(100, 103), (1024, 104), (1024, 105)}, see Figure 2. In addition, note that QP, CP and SG do not obtain sparse solutions, whereas ProxFlow does. We have also run ProxFlow and SG on a larger dataset with (n, p) = (100, 106): after 12 hours, ProxFlow and SG have reached a relative duality gap of 0.0006 and 0.02 respectively.7 6 7

In our simulations, we use the commercial software Mosek, http://www.mosek.com/. Due to the computational burden, QP and CP could not be run on every problem.

6

?4 ?6 ?8 ?10 ?2

CP QP ProxFlow SG

?1 0 1 2 log(CPU time) in seconds

n=1024, p=10000, two?dimensional DCT

2 0 ?2 ?4 ?6 ?8 ?10 ?2

CP ProxFlow SG

0 2 log(CPU time) in seconds

4

log(relative distance to optimum)

0 ?2

log(relative distance to optimum)

log(relative distance to optimum)

n=100, p=1000, one?dimensional DCT

2

n=1024, p=100000, one?dimensional DCT

2 0 ?2 ?4 ?6 ?8 ?10 ?2

ProxFlow SG

0 2 log(CPU time) in seconds

4

Figure 2: Speed comparisons: distance to the optimal primal value versus CPU time (log-log scale).6

Figure 3: From left to right: original image y; estimated background Xw; foreground (the sparsity pattern of e used as mask on y) estimated with ?1 ;

foreground estimated with ?1 + ?; another foreground obtained with ?, on a different image, with the same values of ?1 , ?2 as for the previous image. For the top row, the percentage of pixels matching the ground truth is 98.8% with ?, 87.0% without. As for the bottom row, the result is 93.8% with ?, 90.4% without (best seen in color). 4.2

Background Subtraction

Following [9, 10], we consider a background subtraction task. Given a sequence of frames from a fixed camera, we try to segment out foreground objects in a new image. If we denote by y ? Rn a test image, we model y as a sparse linear combination of p other images X ? Rn?p , plus an error term e in Rn , i.e., y ? Xw + e for some sparse vector w in Rp . This approach is reminiscent of [29] in the context of face recognition, where e is further made sparse to deal with occlusions. The term Xw accounts for background parts present in both y and X, while e contains specific, or foreground, objects in y. The resulting optimization problem is minw,e 21 ky ? Xw ? ek22 + ?1 kwk1 + ?2 kek1 , with ?1 , ?2 ? 0. In this formulation, the ?1 -norm penalty on e does not take into account the fact that neighboring pixels in y are likely to share the same label (background or foreground), which may lead to scattered pieces of foreground and background regions (Figure 3). We therefore put an additional structured regularization term ? on e, where the groups in G are all the overlapping 3?3-squares on the image. A dataset with hand-segmented evaluation images is used to illustrate the effect of ?.8 For simplicity, we use a single regularization parameter, i.e., ?1 = ?2 , chosen to maximize the number of pixels matching the ground truth. We consider p = 200 images with n = 57600 pixels (i.e., a resolution of 120?160, times 3 for the RGB channels). As shown in Figure 3, adding ? improves the background subtraction results for the two tested videos, by encoding, unlike the ?1 -norm, both spatial and color consistency. 4.3

Multi-Task Learning of Hierarchical Structures

In [12], Jenatton et al. have recently proposed to use a hierarchical structured norm to learn dictionaries of natural image patches. Following this work, we seek to represent n signals {y1 , . . . , yn } of dimension m as sparse linear combinations of elements from a dictionary X = [x1 , . . . , xp] in Rm?p . This can be expressed for all i in [1; n] as yi ? Xwi , for some sparse vector wi in Rp . In [12], the dictionary elements are embedded in a predefined tree T , via a particular instance of the structured norm ?; we refer to it as ?tree , and call G the underlying set of groups. In this case, each signal yi admits a sparse decomposition in the form of a subtree of dictionary elements. 8

http://research.microsoft.com/en-us/um/people/jckrumm/wallflower/testimages.htm
7

Inspired by ideas from multi-task learning [14], we propose to learn the tree structure T by pruning irrelevant parts of a larger initial tree T0 . We achieve this by using an additional regularization term ?joint across the different decompositions, so that subtrees of T0 will simultaneously be removed for all signals yi . In other words, the approach of [12] is extended by the following formulation: n i 1 Xh 1 i ky ? Xwi k22 + ?1 ?tree (wi) +?2 ?joint (W), s.t. kxj k2 ? 1, for all j in [1; p], (6) min X,W n 2 i=1 where W , [w1 , . . . , wn] is the matrix of decomposition coefficients in Rp?n . The new regularP ization term operates on the rows of W and is defined as ?joint (W) , g?G maxi?[1;n] —wgi —.9 The overall penalty on W, which results from the combination of ?tree and ?joint , is itself an instance of ? with general overlapping groups, as defined in Eq (2).

Mean Square Error

To address problem (6), we use the same optimization scheme as [12], i.e., alternating between X and W, fixing one variable while optimizing with respect to the other. The task we consider is the denoising of natural image patches, with the same dataset and protocol as [12]. We study whether learning the hierarchy of the dictionary elements improves the denoising performance, compared to standard sparse coding (i.e., when ?tree is the ?1 -norm and ?2 = 0) and the hierarchical dictionary learning of [12] based on predefined trees (i.e., ?2 = 0). The dimensions of the training set ? 50 000 patches of size 8?8 for dictionaries with up to p = 400 elements ? impose to handle large graphs, with —E— ? —V — ? 4.107 . Since problem (6) is too large to be solved many times to select the regularization parameters (?1 , ?2) rigorously, we use the following heuristics: we optimize mostly with the currently pruned tree held fixed (i.e., ?2 = 0), and only prune the tree (i.e., ?2 ¿ 0) every few steps on a random subset of 10 000 patches. We consider the same hierarchies as in [12], involving between 30 and 400 dictionary elements. The regularization parameter ?1 is selected on the validation set of 25 000 patches, for both sparse coding (Flat) and hierarchical dictionary learning (Tree). Starting from the tree giving the best performance (in this case the largest one, see Figure 4), we solve problem (6) following our heuristics, for increasing values of ?2 . As shown in Figure 4, there is a regime where our approach performs significantly better than the two other compared methods. The standard deviation of the noise is 0.2 (the pixels have values in [0, 1]); no significant improvements were observed for lower levels of noise. Denoising Experiment: Mean Square Error 0.21 Flat Tree Multi?task Tree 0.2

0.19 0

100 200 300 Dictionary Size

400

Figure 4: Left: Hierarchy obtained by pruning a larger tree of 76 elements. Right: Mean square error versus dictionary size. The error bars represent two standard deviations, based on three runs.

5

Conclusion

We have presented a new optimization framework for solving sparse structured problems involving sums of ?? -norms of any (overlapping) groups of variables. Interestingly, this sheds new light on connections between sparse methods and the literature of network flow optimization. In particular, the proximal operator for the formulation we consider can be cast as a quadratic min-cost flow problem, for which we propose an efficient and simple algorithm. This allows the use of accelerated gradient methods. Several experiments demonstrate that our algorithm can be applied to a wide class of learning problems, which have not been addressed before within sparse methods. Acknowledgments This paper was partially supported by the European Research Council (SIERRA Project). The authors would like to thank Jean Ponce for interesting discussions and suggestions. 9

The simplified case where ?tree and ?joint are the ?1 - and mixed ?1 /?2 -norms [13] corresponds to [30].

8

2 References

[1] P. Bickel, Y. Ritov, and A. Tsybakov. Simultaneous analysis of Lasso and Dantzig selector. Ann. Stat., 37(4):1705?1732, 2009. [2] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. Ann. Stat., 32(2):407?499, 2004. [3] Y. Nesterov. Gradient methods for minimizing composite objective function. Technical report, Center for Operations Research and Econometrics (CORE), Catholic University of Louvain, 2007. [4] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imag. Sci., 2(1):183?202, 2009. [5] R. Jenatton, J-Y. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. Technical report, 2009. Preprint arXiv:0904.3523v1. [6] L. Jacob, G. Obozinski, and J.-P. Vert. Group Lasso with overlap and graph Lasso. In Proc. ICML, 2009. [7] P. Zhao, G. Rocha, and B. Yu. The composite absolute penalties family for grouped and hierarchical variable selection. Ann. Stat., 37(6A):3468?3497, 2009. [8] R. G. Baraniuk, V. Cevher, M. Duarte, and C. Hegde. Model-based compressive sensing. IEEE T. Inform. Theory, 2010. to appear. [9] V. Cehver, M.F. Duarte, C. Hedge, and R.G. Baraniuk. Sparse signal recovery using markov random fields. In Adv. NIPS, 2008. [10] J. Huang, T. Zhang, and D. Metaxas. Learning with structured sparsity. In Proc. ICML, 2009. [11] S. Kim and E. P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In Proc. ICML, 2010. [12] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for sparse hierarchical dictionary learning. In Proc. ICML, 2010. [13] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. J. Roy. Stat. Soc. B, 68:49?67, 2006. [14] G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. Stat. Comput., 20(2):231?252, 2010. [15] F. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In Adv. NIPS, 2008. [16] P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In Fixed-Point Algorithms for Inverse Problems in Science and Engineering. Springer, 2010. [17] J. Mairal, R. Jenatton, G. Obozinski, and F. Bach. Network flow algorithms for structured sparsity. Technical report, 2010. Preprint arXiv:1008.5209v1. [18] D. S. Hochbaum and S. P. Hong. About strongly polynomial time algorithms for quadratic optimization over submodular constraints. Math. Program., 69(1):269?309, 1995. [19] P. Brucker. An O(n) algorithm for quadratic knapsack problems. Oper. Res. Lett., 3:163?166,

1984. [20] G. Gallo, M. E. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. SIAM J. Comput., 18:30?55, 1989. [21] M. Babenko and A.V. Goldberg. Experimental evaluation of a parametric flow algorithm. Technical report, Microsoft Research, 2006. MSR-TR-2006-77. [22] H. Groenevelt. Two algorithms for maximizing a separable concave function over a polymatroid feasible region. Eur. J. Oper. Res., pages 227?236, 1991. [23] F. Bach. Structured sparsity-inducing norms through submodular functions. In Adv. NIPS, 2010. [24] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. In Proc. of ACM Symposium on Theory of Computing, 1986. [25] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. Canadian J. Math., 8(3), 1956. [26] B. V. Cherkassky and A. V. Goldberg. On implementing the pushrelabel method for the maximum flow problem. Algorithmica, 19(4):390?410, 1997. [27] S. Negahban, P. Ravikumar, M. J. Wainwright, and B. Yu. A unified framework for high-dimensional analysis of M-estimators with decomposable regularizers. In Adv. NIPS, 2009. [28] J. M. Borwein and A. S. Lewis. Convex analysis and nonlinear optimization. Springer, 2006. [29] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. IEEE T. Pattern. Anal., pages 210?227, 2008. [30] P. Sprechmann, I. Ramirez, G. Sapiro, and Y. C. Eldar. Collaborative hierarchical sparse modeling. Technical report, 2010. Preprint arXiv:1003.0400v1.

9