Differentiable Learning of Submodular Models

Josip Djolonga
Department of Computer Science
ETH Zurich
josipd@inf.ethz.ch

Andreas Krause
Department of Computer Science
ETH Zurich
krausea@ethz.ch

Abstract

Can we incorporate discrete optimization algorithms within modern machine learning models? For example, is it possible to incorporate in deep architectures a layer whose output is the minimal cut of a parametrized graph? Given that these models are trained end-to-end by leveraging gradient information, the introduction of such layers seems very challenging due to their non-continuous output. In this paper we focus on the problem of submodular minimization, for which we show that such layers are indeed possible. The key idea is that we can continuously relax the output without sacrificing guarantees. We provide an easily computable approximation to the Jacobian complemented with a complete theoretical analysis. Finally, these contributions let us experimentally learn probabilistic log-supermodular models via a bi-level variational inference formulation.

1 Introduction

Discrete optimization problems are ubiquitous in machine learning. While the majority of them are provably hard, a commonly exploitable trait that renders some of them tractable is that of submodularity [1, 2]. In addition to capturing many useful phenomena, submodular functions can be minimized in polynomial time and also enjoy a powerful connection to convex optimization [3]. Both of these properties have been used to great effect in both computer vision and machine learning, to e.g. compute the MAP configuration in undirected graphical models with long-reaching interactions [4] and higher-order factors [5], clustering [6], to perform variational inference in log-supermodular models [7, 8], or to design norms useful for structured sparsity problems [9, 10].

Despite all the benefits of submodular functions, the question of how to learn them in a practical manner remains open. Moreover, if we want to open the toolbox of submodular optimization to modern practitioners, an intriguing question is how to to use them in conjunction with deep learning networks. For instance, we need to develop mechanisms that would enable them to be trained together in a fully end-to-end fashion. As a concrete example from the computer vision domain, consider the problem of image segmentation. Namely, we are given as input an RGB representation $\mathbf{x} \in \mathbb{R}^{n\times 3}$ of an image captured by say a dashboard camera, and the goal is to identify the set of pixels $A \subseteq \{1, 2, \ldots, n\}$ that are occupied by pedestrians. While we could train a network $\boldsymbol{\theta}\colon \mathbf{x} \to \mathbf{v} \in \mathbb{R}^n$ to output per-pixel scores, it would be helpful, especially in domains with limited data, to bias the predictions by encoding some prior beliefs about the expected output. For example, we might prefer segmentations that are spatially consistent. One common approach to encourage such configurations is to first define a graph over the image $\mathcal{G} = (V, E)$ by connecting neighbouring pixels, specify weights $\mathbf{w}$ over the edges, and then solve the following graph-cut problem

$$A^*(\mathbf{w}, \mathbf{v}) = \operatorname*{arg\,min}_{A\subseteq V} F(A) = \operatorname*{arg\,min}_{A\subseteq V} \sum_{\{i,j\}\in E} w_{i,j} \underbrace{[\![A \cap \{i,j\} = 1]\!]}_{\text{1 iff the predictions disagree}} + \sum_{i\in A} \underbrace{v_i}_{\text{pixel score}} . \quad (1)$$

While this can be easily seen as a module computing the best configuration as a function of the edge weights and per-pixel scores, incorporating it as a layer in a deep network seems at a first glance to

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

be a futile task. Even though the output is easily computable, it will be discontinuous and have no Jacobian, which is necessary for backpropagation. However, as the above problem is an instance of submodular minimization, we can leverage its relationship to convexity and relax it to

$$\mathbf{y}^*(\mathbf{w}, \mathbf{v}) = \underset{\mathbf{y} \in \mathbb{R}^n}{\arg\min}\, f(\mathbf{y}) + \frac{1}{2}\|\mathbf{y}\|^2 = \underset{\mathbf{y} \in \mathbb{R}^n}{\arg\min} \sum_{\{i,j\} \in E} w_{i,j}|y_i - y_j| + \mathbf{v}^T \mathbf{y} + \frac{1}{2}\|\mathbf{y}\|^2. \quad (2)$$

In addition to having a continuous output, this relaxation has a very strong connection with the discrete problem as the discrete optimizer can be obtained by thresholding $\mathbf{y}^*$ as $A^* = \{i \in V \mid y_i^* > 0\}$. Moreover, as explained in §2, for every submodular function F there exists an easily computable convex function f so that this relationship holds. For general submodular functions, the negation of the solution to the relaxed problem (2) is known as the *min-norm point* [11]. In this paper we consider the problem of designing such modules that solve discrete optimization problems by leveraging this continuous formulation. To this end, our key technical contribution is to analyze the sensitivity of the min-norm point as a function of the parametrization of the function f. For the specific case above we will show how to compute $\partial \mathbf{y}^*/\partial \mathbf{w}$ and $\partial \mathbf{y}^*/\partial \mathbf{v}$.

Continuing with the segmentation example, we might want to train a conditional model $P(A \mid \mathbf{x})$ that can model the uncertainty in the predictions to be used in downstream decision making. A rich class of models are log-supermodular models, i.e., those of the form $P(A \mid \mathbf{x}) = \exp(-F_{\boldsymbol{\theta}(\mathbf{x})}(A))/\mathcal{Z}(\boldsymbol{\theta})$ for some parametric submodular function $F_{\boldsymbol{\theta}}$. While they can capture very useful interactions, they are very hard to train in the maximum likelihood setting due to the presence of the intractable normalizer $\mathcal{Z}(\boldsymbol{\theta})$. However, Djolonga and Krause [8] have shown that for any such distribution we can find the closest fully factorized distribution $Q(\cdot \mid \mathbf{x})$ minimizing a specific information theoretic divergence D_∞. In other words, we can *exactly* compute $Q(\cdot \mid \mathbf{x}) = \arg\min_{Q \in \mathcal{Q}} D_\infty(P(\cdot \mid \mathbf{x}) \,\|\, Q)$, where $\mathcal{Q}$ is the family of fully factorized distributions. Most importantly, the optimal Q can also be computed from the min-norm point. Thus, a reasonable objective would be to learn a model $\boldsymbol{\theta}(\mathbf{x})$ so that the best *approximate* distribution $Q(\cdot \mid \mathbf{x})$ gives high likelihood to the training data point. This is a complicated bi-level optimization problem (as Q implicitly depends on $\boldsymbol{\theta}$) with an inner variational inference procedure, which we can again train end-to-end using our results. In other words, we can optimize the following algorithm end-to-end with respect to $\boldsymbol{\theta}$.

$$\mathbf{x}_i \xrightarrow{\theta(\mathbf{x})} \boldsymbol{\theta}_i \longrightarrow P = \exp(-F_{\boldsymbol{\theta}_i}(A))/\mathcal{Z}(\boldsymbol{\theta}_i) \longrightarrow Q = \underset{Q \in \mathcal{Q}}{\arg\min}\, D_\infty(P \,\|\, Q) \longrightarrow Q(A_i \mid \mathbf{x}_i). \quad (3)$$

Related work. Sensitivity analysis of the set of optimal solutions has a long history in optimization theory [12]. The problem of argmin-differentiation of the specific case resulting from graph cuts (i.e. eq. (2)) has been considered in the computer vision literature, either by smoothing the objective [13], or by unrolling iterative methods [14]. The idea to train probabilistic models by evaluating them using the marginals produced by an approximate inference algorithm has been studied by Domke [15] for tree-reweighted belief propagation and mean field, and for continuous models by Tappen [16]. These methods either use the implicit function theorem, or unroll iterative optimization algorithms. The benefits of using an inconsistent estimator, which is what we do by optimizing eq. (3), at the benefit of using computationally tractable inference methods has been discussed by Wainwright [17]. Amos and Kolter [18] discuss how to efficiently argmin-differentiate quadratic programs by perturbing the KKT conditions, an idea that goes back to Boot [19]. We make an explicit connection to their work in Theorem 4. In Section 4 we harness the connection between the min-norm problem and isotonic regression, which has been exploited to obtain better duality certificates [2], and by Kumar and Bach [20] to design an active-set algorithm for the min-norm problem. Chakravarti [21] analyzes the sensitivity of the optimal isotonic regression point with respect to perturbations of the input, but does not discuss the directional derivatives of the problem. Recently, Dolhansky and Bilmes [22] have used deep networks to parametrize submodular functions. Discrete optimization is also used in structured prediction [23, 24] for the computation of the loss function, which is closely related to our work if we use discrete optimization only at the last layer. However, in this case we have the advantage in that we allow for arbitrary loss functions to be applied to the solution of the relaxation.

Contributions. We develop a very efficient approximate method (§4) for the computation of the Jacobian of the min-norm problem inspired by our analysis of isotonic regression in §3, where we derive results that might be of independent interest. Even more importantly, from a practical perspective, this Jacobian has a very nice structure and we can multiply with it in linear time. This means that we can efficiently perform back-propagation if we use these layers in a modern deep architectures. In §5 we show how to compute directional derivatives exactly in polynomial time, and give conditions under which our approximation is correct. This is also an interesting theoretical result as it quantifies the stability of the min-norm point with respect to the model parameters. Lastly, we use our results to learn log-supermodular models in §6.

2 Background on Submodular Minimization

Let us introduce the necessary background on submodular functions. They are defined over subsets of some *ground set*, which in the remaining of this paper we will w.l.o.g. assume to be $V = \{1, 2, \ldots, n\}$. Then, a function $F\colon 2^V \to \mathbb{R}$ is said to be submodular iff for all $A, B \subseteq V$ it holds that

$$F(A \cup B) + F(A \cap B) \leq F(A) + F(B). \tag{4}$$

We will furthermore w.l.o.g. assume that F is normalized so that $F(\emptyset) = 0$. A very simple family of submodular functions are modular functions. These, seen as discrete analogues of linear functions, satisfy the above with equality and are given as $F(A) = \sum_{i \in A} m_i$ for some real numbers m_i. As common practice in combinatorial optimization, we will treat any vector $\mathbf{m} \in \mathbb{R}^n$ as a modular function $2^V \to \mathbb{R}$ defined as $m(A) = \sum_{i \in A} m_i$. In addition to the graph cuts from the introduction (eq. (1)), another widely used class of functions are concave-of-cardinality functions, i.e. those of the form $F(|A|) = h(|A|)$ for some concave $h\colon \mathbb{R} \to \mathbb{R}$ [5]. From eq. (4) we see that if we want to define a submodular function over a collection $\mathcal{D} \subsetneq 2^V$ it has to be closed under union and intersection. Such collections are known as *lattices*, and two examples that we will use are the simple lattice 2^V and the trivial lattice $\{\emptyset, V\}$. In the theory of submodular minimization, a critical object defined by a pair consisting of a submodular function F and a lattice $\mathcal{D} \supseteq \{\emptyset, V\}$ is the base polytope

$$\mathcal{B}(F \mid \mathcal{D}) = \{\mathbf{x} \in \mathbb{R}^n \mid x(A) \leq F(A) \text{ for all } A \in \mathcal{D}\} \cap \{\mathbf{x} \in \mathbb{R}^n \mid x(V) = F(V)\}. \tag{5}$$

We will also use the shorthand $\mathcal{B}(F) = \mathcal{B}(F \mid 2^V)$. Using the result of Edmonds [25], we know how to maximize a linear function over $\mathcal{B}(F)$ in $O(n \log n)$ time with n function evaluations of F. Specifically, to compute $\max_{\mathbf{y} \in \mathcal{B}(F)} \mathbf{z}^T \mathbf{y}$, we first choose a permutation $\sigma\colon V \to V$ that sorts $\mathbf{z}$, i.e. so that $z_{\sigma(1)} \geq z_{\sigma(2)} \geq \cdots \geq z_{\sigma(n)}$. Then, a maximizer $\mathbf{f}(\sigma) \in \mathcal{B}(F)$ can be computed as

$$[\mathbf{f}(\sigma)]_{\sigma(i)} = F(\{\sigma(i)\} \mid \{\sigma(1), \ldots, \sigma(i-1)\}), \tag{6}$$

where the marginal gain of A given B is defined as $F(A \mid B) = F(A \cup B) - F(B)$. Hence, we know how to compute the support function $f(\mathbf{z}) = \sup_{\mathbf{y} \in \mathcal{B}(F)} \mathbf{z}^T \mathbf{y}$, which is known as the Lovász extension [3]. First, this function is indeed an extension as $f(\mathbf{1}_A) = F(A)$ for all $A \subseteq V$, where $\mathbf{1}_A \in \{0,1\}^n$ is the indicator vector for the set A. Second, it is convex as it is a supremum of linear functions. Finally, and most importantly, it lets us minimize submodular functions in polynomial time with convex optimization because $\min_{A \in 2^V} F(A) = \min_{\mathbf{z} \in [0,1]} f(\mathbf{z})$ and we can efficiently round the optimal continuous point to a discrete optimizer. Another problem, with a smooth objective, which is also explicitly tied to the problem of minimizing F is that of computing the min-norm point, which can be defined in two different ways as

$$\mathbf{y}^* = \operatorname*{arg\,min}_{\mathbf{y} \in \mathcal{B}(F)} \frac{1}{2}\|\mathbf{y}\|^2, \text{ or equivalently as } -\mathbf{y}^* = \operatorname*{arg\,min}_{\mathbf{y}} f(\mathbf{y}) + \frac{1}{2}\|\mathbf{y}\|^2, \tag{7}$$

where the equivalence comes from strong Fenchel duality [2]. The connection with submodular minimization comes from the following lemma, which we have already hinted at in the introduction.

Lemma 1 ([1, Lem. 7.4])**.** *Define $A_- = \{i \mid y_i^* < 0\}$ and $A_0 = \{i \mid y_i^* \leq 0\}$. Then A_- (A_0) is the unique smallest (largest) minimizer of F.*

Moreover, if instead of hard-thresholding we send the min-norm point through a sigmoid, the result has the following variational inference interpretation, which lets us optimize the pipeline in eq. (3).

Lemma 2 ([8, Thm. 3])**.** *Define the infinite Rényi divergence between any distributions P and Q over 2^V as $D_\infty(P \parallel Q) = \sup_{A \subseteq V} \log\left[P(A)/Q(A)\right]$. For $P(A) \propto \exp(-F(A))$, the distribution Q^* minimizing D_∞ over all fully factorized distributions Q is given as*

$$Q(A) = \prod_{i \in A} \sigma(-y_i^*) \prod_{i \notin A} \sigma(y_i^*),$$

where $\sigma(u) = 1/(1 + \exp(-u))$ is the sigmoid function.

3 Argmin-Differentiation of Isotonic Regression

We will first analyze a simpler problem, i.e., that of isotonic regression, defined as

$$\mathbf{y}(\mathbf{x}) = \arg\min_{\mathbf{y}\in\mathcal{O}} \frac{1}{2}\|\mathbf{y}-\mathbf{x}\|^2, \tag{8}$$

where $\mathcal{O} = \{\mathbf{y} \in \mathbb{R}^n \mid y_i \le y_{i+1} \text{ for } i = 1, 2, \ldots, n-1\}$. The connection to our problem will be made clear in Section 4, and it essentially follows from the fact that the Lovász extension is linear on $\mathcal{O}$. In this section, we will be interested in computing the Jacobian $\partial\mathbf{y}/\partial\mathbf{x}$, i.e., in understanding how the solution $\mathbf{y}$ changes with respect to the input $\mathbf{x}$. The function is well-defined because of the strict convexity of the objective and the non-empty convex feasible set. Moreover, it can be easily computed in $O(n)$ time using the pool adjacent violators algorithm (PAVA) [26]. This is a well-studied problem in statistics, see e.g. [27]. To understand the behaviour of $\mathbf{y}(\mathbf{x})$, we will start by stating the optimality conditions of problem (8). To simplify the notation, for any $A \subseteq V$ we will define $\mathrm{Mean}_{\mathbf{x}}(A) = \frac{1}{|A|}\sum_{i\in A} x_i$. The optimality conditions can be stated via ordered partitions $\Pi = (B_1, B_2, \ldots, B_m)$ of V, meaning that the sets B_i are disjoint, $\cup_{j=1}^k B_j = V$, and Π is ordered so that $1 + \max_{i\in B_j} i = \min_{i\in B_{j+1}} i$. Specifically, for any such partition we define $\mathbf{y}_\Pi = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m)$, where $\mathbf{y}_j = \mathrm{Mean}_{\mathbf{x}}(B_j)\mathbf{1}_{|B_j|}$ and $\mathbf{1}_k = \{1\}^k$ is the vector of all ones. In other words, $\mathbf{y}_\Pi$ is defined by taking block-wise averages of $\mathbf{x}$ with respect to Π. By analyzing the KKT conditions of problem (8), we obtain the following well-known condition.

Lemma 3 ([26]). *An ordered partition $\Pi = (B_1, B_2, \ldots, B_m)$ is optimal iff the following hold*

1. (Primal feasibility) *For any two blocks B_j and B_{j+1} we have*

$$\mathit{Mean}_{\mathbf{x}}(B_j) \le \mathit{Mean}_{\mathbf{x}}(B_{j+1}). \tag{9}$$

2. (Dual feasibility) *For every block $B \in \Pi$ and each $i \in B$ define $\mathit{Pre}_B(i) = \{j \in B \mid j \le i\}$. Then, the condition reads*

$$\mathit{Mean}_{\mathbf{x}}(\mathit{Pre}_B(i)) - \mathit{Mean}_{\mathbf{x}}(B) \ge 0. \tag{10}$$

Points where eq. (9) is satisfied with equality are of special interest, because of the following result.

Lemma 4. *If for some B_j and B_{j+1} the first condition is satisfied with equality, we can merge the two sets so that the new coarser partition Π' will also be optimal.*

Thus, in the remaining of this section we will assume that the sets B_j are chosen maximally. We will now introduce a notion that will be crucial in the subsequent analysis.

Definition 1. *For any block B, we say that $i \in B$ is a* breakpoint *if $\mathit{Mean}_{\mathbf{x}}(\mathit{Pre}_B(i)) = \mathit{Mean}_{\mathbf{x}}(B)$ and it is not the right end-point of B (i.e., $i < \max_{i'\in B} i'$).*

From an optimization perspective, any breakpoint is equivalent to non-strict complementariness of the corresponding Lagrange multiplier. From a combinatorial perspective, they correspond to positions where we can refine Π into a finer partition Π' that gives rise to the same point, i.e., $\mathbf{y}_\Pi = \mathbf{y}_{\Pi'}$ (if we merge blocks using Lemma 4, the point where we merge them will become a breakpoint). We can now discuss the differentiability of $\mathbf{y}(\mathbf{x})$. Because projecting onto convex sets is a proximal operator and thus non-expansive, we have the following as an immediate consequence of Rademacher's theorem.

Lemma 5. *The function $\mathbf{y}(\mathbf{x})$ is 1-Lipschitz continuous and differentiable almost everywhere.*

We will denote by $\partial^-_{x_k}$ and $\partial^+_{x_k}$ the left and right partial derivatives with respect to x_k. For any index k we will denote by $u(k)$ ($l(k)$) the breakpoint with the smallest (largest) coordinate larger (smaller) than k. Define it as $+\infty$ ($-\infty$) if no such point exists. Moreover, denote by $\Pi(\mathbf{z})$ the collection of indices where $\mathbf{z}$ takes on distinct values, i.e., $\Pi(\mathbf{z}) = \cup_{i=1}^n \{\{i' \in V \mid z_i = z_{i'}\}\}$.

Theorem 1. *Let k be any coordinate and let $B \in \Pi(\mathbf{y}(\mathbf{x}))$ be the block containing coordinate i. Also define $B_+ = \{i \in B \mid i \ge u(k)\}$ and $B_- = \{i \in B \mid i \le l(k)\}$. Hence, for any $i \in B$*

$$\partial^+_{x_k}(y_i) = [\![i \in B \setminus B_-]\!]/|B \setminus B_-|, \quad \textit{and} \quad \partial^-_{x_k}(y_i) = [\![i \in B \setminus B_+]\!]/|B \setminus B_+|.$$

Note that all of these derivatives will agree iff there are no breakpoints, which means that the existence of breakpoints is an isolated phenomenon due to Lemma 5. In this case the Jacobian exists and has a very simple block-diagonal form. Namely, it is equal to

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \Lambda(\mathbf{y}(\mathbf{x})) \equiv \texttt{blkdiag}(C_{|B_1|}, C_{|B_2|}, \ldots, C_{|B_m|}), \tag{11}$$

where $C_k = \mathbf{1}_{k\times k}/k$ is the averaging matrix with elements $1/k$. We will use $\Lambda(\mathbf{z})$ for the matrix taking block-wise averages with respect to the blocks $\Pi(\mathbf{z})$. As promised in the introduction, Jacobian multiplication $\Lambda(\mathbf{y}(\mathbf{x}))\mathbf{u}$ is linear as we only have to perform block-wise averages.

4 Min-Norm Differentiation

In this section we will assume that we have a function $F_{\boldsymbol{\theta}}$ parametrized by some $\boldsymbol{\theta} \in \mathbb{R}^d$ that we seek to learn. For example, we could have a mixture model

$$F_{\boldsymbol{\theta}}(A) = \sum_{j=1}^{d} \theta_j G_j(A), \tag{12}$$

for some fixed submodular functions $G_j \colon 2^V \to \mathbb{R}$. In this case, to ensure that the resulting function is submodular we also want to enforce $\theta_j \geq 0$ unless G_j is modular. We would like to note that the discussion in this section goes beyond such models. Remember that the min-norm point is defined as

$$\mathbf{y}_{\boldsymbol{\theta}} = -\arg\min_{\mathbf{y}} f_{\boldsymbol{\theta}}(\mathbf{y}) + \frac{1}{2}\|\mathbf{y}\|^2, \tag{13}$$

where $f_{\boldsymbol{\theta}}$ is the Lovász extension of $F_{\boldsymbol{\theta}}$. Hence, we want to compute $\partial \mathbf{y}/\partial \boldsymbol{\theta}$. To make the connection with isotonic regression, remember how we evaluate the Lovász extension at $\mathbf{y}$. First, we pick a permutation σ that sorts $\mathbf{y}$, and then evaluate it as $f_{\boldsymbol{\theta}}(\mathbf{y}) = \mathbf{f}_{\boldsymbol{\theta}}(\sigma)^T\mathbf{y}$, where $\mathbf{f}_{\boldsymbol{\theta}}(\sigma)$ is defined in eq. (6). Hence, the Lovász extension is linear on the set of all vectors that are sorted by σ. Formally, for any permutation σ the Lovász extension is equal to $\mathbf{f}_{\boldsymbol{\theta}}(\sigma)^T\mathbf{y}$ on the *order cone*

$$\mathcal{O}(\sigma) = \{\mathbf{y} \mid y_{\sigma(n)} \leq y_{\sigma(n-1)} \leq \ldots \leq y_{\sigma(1)}\}.$$

Given a permutation σ, if we constrain eq. (13) to $\mathcal{O}(\sigma)$ we can replace $f_\theta(\mathbf{y})$ by the linear function $\mathbf{f}_\theta(\sigma)^T$, so that the problem reduces to

$$\mathbf{y}_{\boldsymbol{\theta}}(\sigma) = -\arg\min_{\mathbf{y}\in\mathcal{O}(\sigma)} \frac{1}{2}\|\mathbf{y} + \mathbf{f}_{\boldsymbol{\theta}}(\sigma)\|^2, \tag{14}$$

which is an instance of isotonic regression if we relabel the elements of V using σ. Roughly, the idea is to instead differentiate eq. (14) with $\mathbf{f}_{\boldsymbol{\theta}}(\sigma)$ computed at the optimal point $\mathbf{y}_{\boldsymbol{\theta}}$. However, because we can choose an arbitrary order among the elements with equal values, there may be multiple permutations that sort $\mathbf{y}_{\boldsymbol{\theta}}$, and this extra choice we have seems very problematic. Nevertheless, let us continue with this strategy and analyze the resulting approximations to the Jacobian. We propose the following approximation to the Jacobian

$$\frac{\partial \mathbf{y}_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} \approx \widehat{J}_\sigma \equiv \underbrace{\Lambda(\mathbf{y}_{\boldsymbol{\theta}})}_{\approx \frac{\partial \mathbf{y}_{\boldsymbol{\theta}}(\sigma)}{\partial \mathbf{f}_{\boldsymbol{\theta}}(\sigma)}} \times \frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\sigma)}{\partial \boldsymbol{\theta}} = \Lambda(\mathbf{y}_{\boldsymbol{\theta}}) \times [\partial_{\theta_1}\mathbf{f}_{\boldsymbol{\theta}}(\sigma) \quad | \quad \partial_{\theta_2}\mathbf{f}_{\boldsymbol{\theta}}(\sigma) \quad | \quad \cdots \quad | \quad \partial_{\theta_d}\mathbf{f}_{\boldsymbol{\theta}}(\sigma)],$$

where $\Lambda(\mathbf{y}_{\boldsymbol{\theta}})$ is used as an approximation of a Jacobian which might not exist. Fortunately, due to the special structure of the linearizations, we have the following result that the gradient obtained using the above strategy *does not depend* on the specific permutation σ that was chosen.

Theorem 2. *If $\partial_{\theta_k} F(A)$ exists for all $A \subseteq V$ the approximate Jacobians $\widehat{J}_\sigma$ are equal and do not depend on the choice of σ. Specifically, the j-th block of any element $i \in B \in \Pi(\mathbf{y}_{\boldsymbol{\theta}})$ is equal to*

$$\frac{1}{|B|}\partial_{\theta_j} F_{\boldsymbol{\theta}}(B \mid \{i' \mid [\mathbf{y}_{\boldsymbol{\theta}}]_{i'} < [\mathbf{y}_{\boldsymbol{\theta}}]_i\}). \tag{15}$$

Proof sketch, details in supplement. Remember that $\Lambda(\mathbf{y}_{\boldsymbol{\theta}})$ averages $\mathbf{f}_{\boldsymbol{\theta}}(\sigma)$ within each $B \in \Pi(\mathbf{y}_{\boldsymbol{\theta}})$. Moreover, as σ sorts $\mathbf{y}_{\boldsymbol{\theta}}$, the elements in B must be placed consecutively. The coordinates of $\mathbf{f}_{\boldsymbol{\theta}}(\sigma)$ are marginal gains (6) and they will telescope inside the mean, which yields the claimed quantity. □

Graph cuts. As a special, but important case, let us analyze how the approximate Jacobian looks like for a cut function (eq. (1)), in which case eq. (13) reduces to eq. (2). Let $\Pi(\mathbf{y}(\mathbf{w}, \mathbf{v})) = (B_1, B_2, \ldots, B_m)$. For any element $i \in V$ we will denote by $\eta(i) \in \{1, 2, \ldots, m\}$ the index of the block where it belongs to. Then, the approximate Jacobian $\widehat{J}$ at $\boldsymbol{\theta} = (\mathbf{w}, \mathbf{v})$ has entries

$$\widehat{\partial}_{v_j}(y_i) = [\![\eta(i) = \eta(j)]\!]/|B_{\eta(i)}|, \text{ and}$$
$$\widehat{\partial}_{w_{i,j}}(y_k) = \begin{cases} \operatorname{sign}(y_i - y_j)\frac{1}{|B_{\eta(k)}|} & \text{if } \eta(k) = \eta(i), \text{ or} \\ \operatorname{sign}(y_j - y_i)\frac{1}{|B_{\eta(k)}|} & \text{if } \eta(k) = \eta(j), \text{ and} \\ 0 & \text{otherwise,} \end{cases}$$

where the sign function is defined to be zero if the argument is zero. Intuitively, increasing the modular term v_i by δ will increase all the coordinates B in $\mathbf{y}$ that are in the same segment by $\delta/|B|$. On the other hand, increasing the weight of an edge $w_{i,j}$ will have no effect if i and j are already on $\mathbf{y}$ in the same segment, and otherwise it will pull the segments containing i and j together by increasing the smaller one and decreasing the larger one. In the supplementary we provide a `pytorch` module that executes the back propagation pass in $O(|V| + |E|)$ time in about 10 lines of code, and we also derive the approximate Jacobians for concave-of-cardinality and facility location functions.

5 Analysis

We will now theoretically analyze the conditions under which our approximation is correct, and then give a characterization of the *exact* directional derivative together with a polynomial algorithm that computes it. The first notion that will have implications for our analysis is that of (in)separability.

Definition 2. *The function $F\colon 2^V \to \mathbb{R}$ is said to be separable if there exists some $B \subseteq V$ such that $B \notin \{\emptyset, V\}$ and $F(V) = F(B) + F(V \setminus B)$.*

The term separable is indeed appropriate as it implies that $F(A) = F(A \cap B) + F((V \setminus B) \cap A)$ for all $A \subseteq V$ [2, Prop. 4.3], i.e., the function splits as a sum of two functions on disjoint domains. Hence, we can split the problem into two (on B and $V \setminus B$) and analyze them independently. We would like to point out that separability is checkable in cubic time using the algorithm of Queyranne [28]. To simplify the notation, we will assume that we want to compute the derivative at point $\boldsymbol{\theta}' \in \mathbb{R}^d$ which results in the min-norm point $\mathbf{y}' = \mathbf{y}_{\boldsymbol{\theta}} \in \mathbb{R}^n$. We will furthermore assume that $\mathbf{y}'$ takes on unique values $\gamma_1 < \gamma_2 < \cdots < \gamma_k$ on sets $B_1, B_2, \ldots, B_k$ respectively, and we will define the chain $\emptyset = A_0 \subseteq A_1 \subseteq A_2 \subseteq \cdots \subseteq A_k = V$ by $A_j = \cup_{j'=1}^{j} B_{j'}$. A central role in the analysis will be played by the set of constraints in $\mathcal{B}(F_{\boldsymbol{\theta}})$ (see (5)) that are active at $\mathbf{y}_{\boldsymbol{\theta}}$, which makes sense given that we expect small perturbations in $\boldsymbol{\theta}'$ to result in small changes in $\mathbf{y}_{\boldsymbol{\theta}'}$ as well.

Definition 3. *For any submodular function $F\colon 2^V \to \mathbb{R}$ and any point $\mathbf{z} \in \mathcal{B}(F)$ we shall denote by $\mathcal{D}_F(\mathbf{z})$ the lattice of tight sets of $\mathbf{z}$ on $\mathcal{B}(F)$, i.e.*

$$\mathcal{D}_F(\mathbf{z}) = \{A \subseteq V \mid z(A) = F(A)\}.$$

The fact that the above set is indeed a lattice is well-known [1]. Moreover, note that $\mathcal{D}_F(\mathbf{z}) \supseteq \{\emptyset, V\}$. We will also define $\mathcal{D}' = \mathcal{D}_{F_{\boldsymbol{\theta}'}}(\mathbf{y}')$, i.e., the lattice of tight sets at the min-norm point.

5.1 When will the approximate approach work?

We will analyze sufficient conditions so that irrespective of the choice of σ, the isotonic regression problem eq. (14) has no breakpoints, and the left and right derivatives agree. To this end, for any $j \in \{1, 2, \ldots, k\}$ we define the submodular function $F_j\colon 2^{B_j}$ as $F_j(H) = F_{\boldsymbol{\theta}'}(H \mid A_{j-1})$, where we have dropped the dependence on $\boldsymbol{\theta}'$ as it will remain fixed throughout this section.

Theorem 3. *The approximate problem* (14) *is argmin-continuously differentiable irrespective of the chosen permutation σ sorting $\mathbf{y}_{\boldsymbol{\theta}}$ if and only if any of the following equivalent conditions hold.*

- *(a)* $\arg\min_{H \in B_j} \left[F_j(H) - F_j(B_j)|H|/|B_j|\right] = \{\emptyset, B_j\}$.
- *(b)* $\mathbf{y}'_{B_j} \in$ *relint*$(\mathcal{B}(F_j))$*, i.e.* $\mathcal{D}_{F_j}(\mathbf{y}'_{B_j}) = \{\emptyset, B_j\}$*, which is only possible if F_j is inseparable.*

In other words, we can equivalently say that the optimum has to lie on the interior of the face. Moreover, if $\boldsymbol{\theta} \to \mathbf{y}_{\boldsymbol{\theta}}$ is continuous[1], this result implies that the min-norm point is locally defined as averaging within the same blocks using (15), so that the approximate Jacobian is exact.

We would like to point out that one can obtain the same derivatives as the ones suggested in §4, if we perturb the KKT conditions, as done by Amos and Kolter [18]. If we use that approach, in addition to the computational challenges, there is the problem of non-uniqueness of the Lagrange multiplier, and moreover, some valid multipliers might be zero for some of the active constraints. This would render the resulting linear system rank deficient, and it is not clear how to proceed. Remember that when we analyzed the isotonic regression problem in §3 we had non-differentiability due to the exactly same reason — zero multipliers for active constraints, which in that case correspond to the breakpoints.

Theorem 4. *For a specific Lagrange multiplier there exists a solution to the perturbed KKT conditions derived by [18] that gives rise to the approximate Jacobians from Section 4. Moreover, this multiplier is unique if the conditions of Theorem 3 are satisfied.*

5.2 Exact computation

Unfortunately, computing the gradients exactly seems very complicated for arbitrary parametrizations $F_{\boldsymbol{\theta}}$, and we will focus our attention to mixture models of the form given in eq. (12). The directions $\mathbf{v}$ where we will compute the directional derivatives will have in general non-negative components v_j, unless F_j is modular. By leveraging the theory of Shapiro [29], and exploiting the structure of both the min-norm point and the polyhedron $\mathcal{B}(F_{\mathbf{v}} \mid \mathcal{D}')$ we obtain at the following result.

Theorem 5. *Assume that $F_{\boldsymbol{\theta}'}$ is inseparable and let $\mathbf{v}$ be any direction so that $F_{\mathbf{v}}$ is submodular. The directional derivative $\partial \mathbf{y}/\partial \theta_j$ at $\boldsymbol{\theta}'$ in direction $\mathbf{v}$ is given by the solution of the following problem.*

$$\begin{aligned} &\underset{\mathbf{d}}{\text{minimize}} \ \frac{1}{2}\|\mathbf{d}\|^2, \\ &\text{subject to } \mathbf{d} \in \mathcal{B}(F_{\mathbf{v}} \mid \mathcal{D}'), \text{ and} \\ &\qquad\qquad d(B_j) = F_{\mathbf{v}}(A_j) \text{ for } j \in \{1, 2, \ldots, k\}. \end{aligned} \tag{16}$$

First, note that this is again a min-norm problem, but now defined over a reduced lattice $\mathcal{D}'$ with k additional equality constraints. Fortunately, due to these additional equalities, we can split the above problem into k *separate min-norm problems*. Namely, for each $j \in \{1, 2, \ldots, k\}$ collect the lattice of tight sets that intersect B_j as $\mathcal{D}'_j = \{H \cap B_j \mid H \in \mathcal{D}'\}$, and define the function $G_j \colon 2^{B_j} \to \mathbb{R}$ as $G_j(A) = F_{\mathbf{v}}(A \mid A_{j-1})$, where note that the parameter vector $\boldsymbol{\theta}$ is taken as the direction $\mathbf{v}$ in which we want to compute the derivative. Then, the block of the optimal solution of problem (16) corresponding to B_j is equal to

$$\mathbf{d}^*_{B_j} = \underset{\mathbf{y}_j \in \mathcal{B}(G_j \mid \mathcal{D}'_j)}{\arg\min} \ \frac{1}{2}\|\mathbf{y}_j\|^2, \tag{17}$$

which is again a min-norm point problem where the base polytope is defined using the lattice $\mathcal{D}'_j$. We can then immediately draw a connection with the results from the previous subsection.

Corollary 1. *If all latices are trivial, the solution of* (17) *agrees with the approximate Jacobian* (15).

How to solve problem (16)**?** Fortunately, the divide-and-conquer algorithm of Groenevelt [30] can be used to find the min-norm point over arbitrary lattices. To do this, we have to compute for each $i \in B_j$ the unique smallest set H_i^* in $\arg\min_{H_j \ni i} F_j(H_j) - y'(H_j)$, which can be done using submodular minimization after applying the reduction of Schrijver [31].

To highlight the difference with the approximation from section 4, let us consider a very simple case.

Lemma 6. *Assume that G_j is equal to $G_j(A) = [\![i \in A]\!]$ for some $i \in B_j$. Then, the directional derivative is equal to $\mathbf{1}_{|D|}/|D|$ where $D = \{i' \mid i \in H^*_{i'}\}$.*

Hence, while the approximate directional derivative would average over all elements in B_j, the true one averages only over a subset $D \subseteq B_j$ and is possibly sparser. Lemma 6 gives us the exact directional derivatives for the graph cuts, as each component G_j will be either a cut function on

[1]For example if the correspondence $\boldsymbol{\theta} \twoheadrightarrow \mathcal{B}(F_{\boldsymbol{\theta}})$ is hemicontinuous due to Berge's theorem.

two vertices, or a function of the form in Lemma 6. In the first case the directional derivative is zero because $\mathbf{0} \in \mathcal{B}(G_j) \subseteq \mathcal{B}(G_j \mid \mathcal{D}'_j)$. In the second case, we can can either solve exactly using Lemma 6 or use a more sophisticated approximation, generalizing the result from [32] — given that F_j is separable over 2^{B_j} iff the graph is disconnected, we can first separate the graph into connected components, and then take averages within each connected component instead of over B_j.

5.3 Structured attention and constraints

Recently, there has been an interest in the design of structured attention mechanisms, which, as we will now show, can be derived and furthermore generalized using the results in this paper. The first mechanism is the *sparsemax* of Martins and Astudillo [33]. It takes as input a vector and projects it to the probability simplex, which is the base polytope corresponding to $G(A) = \min\{|A|, 1\}$. Concurrently with this work, Niculae and Blondel [32] have analyzed the following problem

$$\mathbf{y}^* = \min_{\mathbf{y} \in \mathcal{B}(G)} f(\mathbf{y}) + \frac{1}{2}\|\mathbf{y} - \mathbf{z}\|^2, \tag{18}$$

for the special case when $\mathcal{B}(G)$ is the simplex and f is the Lovász extension of one of two specific submodular functions. We will consider the general case when G can be any concave-of-cardinality function and F is an arbitrary submodular function. Note that, if either $f(\mathbf{y})$ or the constraint were not present in problem (18), we could have simply leveraged the theory we have developed so far to differentiate it. Fortunately, as done by Niculae and Blondel [32], we can utilize the result of Yu [34] to significantly simplify (18). Namely, because projection onto $\mathcal{B}(G)$ preserves the order of the coordinates [35, Lemma 1], we can write the optimal solution $\mathbf{y}^*$ to (18) as

$$\mathbf{y}^* = \min_{x \in \mathcal{B}(G)} \frac{1}{2}\|\mathbf{y} - \mathbf{y}'\|, \text{ where } \mathbf{y}' = \arg\min_{\mathbf{y}} f(\mathbf{y}) + \frac{1}{2}\|\mathbf{y} - \mathbf{z}\|^2.$$

We can hence split problem (18) into two subtasks — first, compute $\mathbf{y}'$ and then project it onto $\mathcal{B}(G)$. As each operation can reduces to a minimum-norm problem, we can differentiate each of them separately, and thus solve (18) by *stacking* two submodular layers one after the other.

6 Experiments

	CNN		CNN+GC	
	Mean	Std. Dev.	Mean	Std. Dev.
Accuracy	0.8103	0.1391	**0.9121**	0.1034
NLL	0.3919	0.1911	**0.2681**	0.2696
# Fg. Objs.	96.9	65.8	**25.3**	30.6

Figure 1: Test set results. We see that incorporating a graph cut solver improves both the accuracy and negative log-likelihood (NLL), while having consistent segmentations with fewer foreground objects.

We consider the image segmentation tasks from the introduction, where we are given an RGB image $\mathbf{x} \in \mathbb{R}^{n \times 3}$ and are supposed to predict those pixels $\mathbf{y} \in \{0, 1\}^n$ containing the foreground object. We used the Weizmann horse segmentation dataset [36], which we split into training, validation and test splits of sizes 180, 50 and 98 respectively. The implementation was done in `pytorch`[2], and to compute the min-norm point we used the algorithm from [37]. To make the problem more challenging, at training time we randomly selected and revealed only 0.1% of the training set labels. We first trained a convolutional neural network with two hidden layers that directly predicts the per-pixel labels, which we refer to as CNN. Then, we added a second model, which we call CNN+GC, that has the same architecture as the first one, but with an additional graph cut layer, whose weights are parametrized by a convolutional neural network with one hidden layer. Details about the architectures can be found in the supplementary. We train the models by maximizing the log-likelihood of the revealed pixels, which corresponds to the variational bi-level strategy (eq. (3)) due to Lemma 2. We trained using SGD, Adagrad [38] and Adam [39], and chose the model with the best validation score. As evident from the results presented in Section 6, adding the discrete layer improves not only the accuracy (after thresholding the marginals at 0.5) and log-likelihood, but it gives more coherent results as it makes predictions with fewer connected components (i.e., foreground objects). Moreover, if we have a look at the predictions themselves in Figure 2, we can observe that the optimization layer not only removes spurious predictions, but there is is also a qualitative difference in the marginals as they are spatially more consistent.

[2]The code will be made available at `https://www.github.com/josipd/nips-17-experiments`.

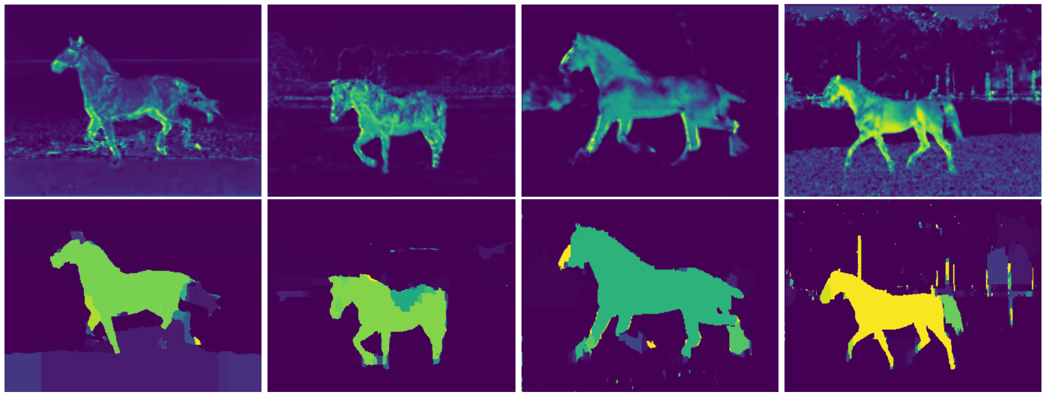

Figure 2: Comparison of results from both models on four instances from the test set (up: CNN, down: CNN+GC). We can see that adding the graph-cut layers helps not only quantitatively, but also qualitatively, as the predictions are more spatially regular and vary smoothly inside the segments.

7 Conclusion

We have analyzed the sensitivity of the min-norm point for parametric submodular functions and provided both a very easy-to-implement practical approximate algorithm for general objectives, and strong theoretical result characterizing the true directional derivatives for mixtures. These results allow the use of submodular minimization inside modern deep architectures, and they are also immediately applicable to bi-level variational learning of log-supermodular models of arbitrarily high order. Moreover, we believe that the theoretical results open the new problem of developing algorithms that can compute not only the min-norm point, but also solve for the associated derivatives.

Acknowledgements. The research was partially supported by ERC StG 307036 and a Google European PhD Fellowship.

References

[1] S. Fujishige. *Submodular functions and optimization*. Annals of Discrete Mathematics vol. 58. 2005.

[2] F. Bach. “Learning with submodular functions: a convex optimization perspective”. *Foundations and Trends® in Machine Learning* 6.2-3 (2013).

[3] L. Lovász. “Submodular functions and convexity”. *Mathematical Programming The State of the Art*. Springer, 1983, pp. 235–257.

[4] Y. Boykov and V. Kolmogorov. “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.9 (2004), pp. 1124–1137.

[5] P. Kohli, L. Ladicky, and P. H. Torr. “Robust higher order potentials for enforcing label consistency”. *Computer Vision and Pattern Recognition (CVPR)*. 2008.

[6] M. Narasimhan, N. Jojic, and J. A. Bilmes. “Q-clustering”. *Advances in Neural Information Processing Systems (NIPS)*. 2006, pp. 979–986.

[7] J. Djolonga and A. Krause. “From MAP to Marginals: Variational Inference in Bayesian Submodular Models”. *Advances in Neural Information Processing Systems (NIPS)*. 2014.

[8] J. Djolonga and A. Krause. “Scalable Variational Inference in Log-supermodular Models”. *International Conference on Machine Learning (ICML)*. 2015.

[9] F. R. Bach. “Shaping level sets with submodular functions”. *Advances in Neural Information Processing Systems (NIPS)*. 2011.

[10] F. R. Bach. “Structured sparsity-inducing norms through submodular functions”. *Advances in Neural Information Processing Systems (NIPS)*. 2010.

[11] S. Fujishige, T. Hayashi, and S. Isotani. *The minimum-norm-point algorithm applied to submodular function minimization and linear programming*. Kyoto University. Research Institute for Mathematical Sciences [RIMS], 2006.

[12] R. T. Rockafellar and R. J.-B. Wets. *Variational analysis*. Vol. 317. Springer Science & Business Media, 2009.

[13] K. Kunisch and T. Pock. "A bilevel optimization approach for parameter learning in variational models". *SIAM Journal on Imaging Sciences* 6.2 (2013), pp. 938–983.

[14] P. Ochs, R. Ranftl, T. Brox, and T. Pock. "Bilevel optimization with nonsmooth lower level problems". *International Conference on Scale Space and Variational Methods in Computer Vision*. Springer. 2015, pp. 654–665.

[15] J. Domke. "Learning graphical model parameters with approximate marginal inference". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.10 (2013), pp. 2454–2467.

[16] M. F. Tappen. "Utilizing variational optimization to learn markov random fields". *Computer Vision and Pattern Recognition (CVPR)*. 2007.

[17] M. J. Wainwright. "Estimating the wrong graphical model: Benefits in the computation-limited setting". *Journal of Machine Learning Research (JMLR)* 7 (2006).

[18] B. Amos and J. Z. Kolter. "OptNet: Differentiable Optimization as a Layer in Neural Networks". *International Conference on Machine Learning (ICML)*. 2017.

[19] J. C. Boot. "On sensitivity analysis in convex quadratic programming problems". *Operations Research* 11.5 (1963), pp. 771–786.

[20] K. Kumar and F. Bach. "Active-set Methods for Submodular Minimization Problems". *arXiv preprint arXiv:1506.02852* (2015).

[21] N. Chakravarti. "Sensitivity analysis in isotonic regression". *Discrete Applied Mathematics* 45.3 (1993), pp. 183–196.

[22] B. Dolhansky and J. Bilmes. "Deep Submodular Functions: Definitions and Learning". *Neural Information Processing Society (NIPS)*. Barcelona, Spain, Dec. 2016.

[23] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. "Large margin methods for structured and interdependent output variables". *Journal of Machine Learning Research (JMLR)* 6.Sep (2005), pp. 1453–1484.

[24] B. Taskar, C. Guestrin, and D. Koller. "Max-margin Markov networks". *Advances in Neural Information Processing Systems (NIPS)*. 2004, pp. 25–32.

[25] J. Edmonds. "Submodular functions, matroids, and certain polyhedra". *Combinatorial structures and their applications* (1970), pp. 69–87.

[26] M. J. Best and N. Chakravarti. "Active set algorithms for isotonic regression; a unifying framework". *Mathematical Programming* 47.1-3 (1990), pp. 425–439.

[27] T. Robertson and T. Robertson. *Order restricted statistical inference*. Tech. rep. 1988.

[28] M. Queyranne. "Minimizing symmetric submodular functions". *Mathematical Programming* 82.1-2 (1998), pp. 3–12.

[29] A. Shapiro. "Sensitivity Analysis of Nonlinear Programs and Differentiability Properties of Metric Projections". *SIAM Journal on Control and Optimization* 26.3 (1988), pp. 628–645.

[30] H. Groenevelt. "Two algorithms for maximizing a separable concave function over a polymatroid feasible region". *European Journal of Operational Research* 54.2 (1991).

[31] A. Schrijver. "A combinatorial algorithm minimizing submodular functions in strongly polynomial time". *Journal of Combinatorial Theory, Series B* 80.2 (2000), pp. 346–355.

[32] V. Niculae and M. Blondel. "A Regularized Framework for Sparse and Structured Neural Attention". *arXiv preprint arXiv:1705.07704* (2017).

[33] A. Martins and R. Astudillo. "From softmax to sparsemax: A sparse model of attention and multi-label classification". *International Conference on Machine Learning (ICML)*. 2016.

[34] Y.-L. Yu. "On decomposing the proximal map". *Advances in Neural Information Processing Systems*. 2013, pp. 91–99.

[35] D. Suehiro, K. Hatano, S. Kijima, E. Takimoto, and K. Nagano. "Online prediction under submodular constraints". *International Conference on Algorithmic Learning Theory*. 2012.

[36] E. Borenstein and S. Ullman. "Combined top-down/bottom-up segmentation". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.12 (2008), pp. 2109–2125.

[37] A. Barbero and S. Sra. “Modular proximal optimization for multidimensional total-variation regularization”. *arXiv preprint arXiv:1411.0589* (2014).

[38] J. Duchi, E. Hazan, and Y. Singer. “Adaptive subgradient methods for online learning and stochastic optimization”. *Journal of Machine Learning Research (JMLR)* 12.Jul (2011), pp. 2121–2159.

[39] D. Kingma and J. Ba. “Adam: A method for stochastic optimization”. *arXiv preprint arXiv:1412.6980* (2014).

A Approximate Jacobians for certain function classes

Cardinality functions. Remember that these functions are of the form $F(A) = h(|A|)$ for some concave h satisfying $h(0) = 0$. Because we only evaluate h at the integers $\{1, 2, \ldots, n\}$, we can parametrize it by $h(k) = \sum_{i=1}^{k} \theta_i$ for some reals $\theta_1 \geq \theta_2 \geq \cdots \geq \theta_n$ (this condition is equivalent to the concavity of h). Then, $\partial_{\theta_k} \mathbf{f}_\sigma$ is equal to one on coordinate $\sigma(k)$ and zero elsewhere. Hence, for any block $B \in \Pi(\mathbf{y}_{\boldsymbol{\theta}})$, the column of the approximate Jacobian corresponding to ∂_{θ_k} will have values that are identical and equal to $[\![\sigma(k) \in B]\!]/|B|$. Again, backpropagation takes linear time.

Facility location. Consider functions of the form $F(A) = \max_{i \in A} w_i$ for some non-negative weights w_i. For any weight w_k the approximate derivative $\widehat{\partial}_{w_k}$ is non-zero only for the block B is containing $\sigma(k)$ and the block B_- succeeding it. It can be again computed in linear time.

B Proofs

Definition 4. *Denote by* $\mathrm{Sym}(V)$ *the set of all permutations* $\sigma\colon V \to V$ *of* V.

Proof of lemma 4. The first condition (eq. (9)) is obviously satisfied after merging, so that we only have to prove that dual feasibility (eq. (10)) will not be violated. Denote the new merged block by $B = B_j \cup B_{j+1}$. First, note that by assumption $\mathrm{Mean}_{\mathbf{x}}(B) = \mathrm{Mean}_{\mathbf{x}}(B_j) = \mathrm{Mean}_{\mathbf{x}}(B_{j+1})$. For any $i \in B_j$ we will have $\mathrm{Mean}_{\mathbf{x}}(\mathrm{Pre}_{B_j}(i)) = \mathrm{Mean}_{\mathbf{x}}(\mathrm{Pre}_B(i))$, so the dual feasibility condition is satisfied from the assumption that the original partition was optimal. On the other hand, for $i \in B_{j+1}$

$$
\begin{aligned}
&\mathrm{Mean}_{\mathbf{x}}(B_j \cup \mathrm{Pre}_{B_{j+1}}(i)) - \mathrm{Mean}_{\mathbf{x}}(B_j \cup B_{j+1}) \\
&= \alpha \mathrm{Mean}_{\mathbf{x}}(B_j) + (1-\alpha)\mathrm{Mean}_{\mathbf{x}}(\mathrm{Pre}_{B_{j+1}}(i)) - \mathrm{Mean}_{\mathbf{x}}(B_j \cup B_{j+1}) \text{ for } \alpha = \frac{|B_j|}{|B_j| + |B_{j+1}|} \\
&= \alpha\big[\mathrm{Mean}_{\mathbf{x}}(B_j) - \mathrm{Mean}_{\mathbf{x}}(B_j)\big] + (1-\alpha)\big[\mathrm{Mean}_{\mathbf{x}}(\mathrm{Pre}_{B_{j+1}}(i)) - \mathrm{Mean}_{\mathbf{x}}(B_{i+1})\big] \\
&\geq 0.
\end{aligned}
$$

□

Proof of theorem 1. We will show how to compute the derivative in direction $\partial_{x_k}^{+}$, as the other case follows analogously. Let us split B into B_- and $B'_+ = B \setminus B_- = \{i_1 + 1, i_1 + 2, \ldots, i_1 + m\}$, and consider the partition Π' formed by replacing B with $\{B_-, B'_+\}$. We will show that the above partition is optimal when we increase x_k by any sufficiently small perturbation $\varepsilon > 0$. Then, the claim immediately follows, because $\mathbf{y}(\mathbf{x})$ is defined by taking averages inside an optimal partition. We thus have to show that both of the conditions in lemma 3 will hold for Π'. The first condition obviously holds as the elements of $\Pi(\mathbf{y}(\mathbf{x}))$ correspond to different values, so there is some non-zero gap between the averages of consecutive blocks. Let us denote by $\mathbf{x}_\varepsilon$ the perturbed point, i.e. $x_{\varepsilon,j} = x_j + [\![j = k]\!]\varepsilon$. Because B_- ends with a breakpoint and $k \notin B_-$, we have that

$$\mathrm{Mean}_{\mathbf{x}_\varepsilon}(B_-) = \mathrm{Mean}_{\mathbf{x}}(B) = \mathrm{Mean}_{\mathbf{x}}(B'_+), \tag{19}$$

which means that the second condition also holds for B_- as none of its points are perturbed, so that the averages stay the same. On the other hand, for any $i \in B'_+$

$$\mathrm{Mean}_{\mathbf{x}}(\mathrm{Pre}_B(i)) = (1-\alpha)\mathrm{Mean}_{\mathbf{x}}(B_-) + \alpha \mathrm{Mean}_{\mathbf{x}}(\mathrm{Pre}_{B'_+}(i)) \text{ for } \alpha = \frac{i - i_1}{|B|} \in (0, 1].$$

Moreover, if we subtract $\mathrm{Mean}_{\mathbf{x}}(B)$ from both sides of the above equality we have (from eq. (19))

$$\mathrm{Mean}_{\mathbf{x}}(\mathrm{Pre}_B(i)) - \mathrm{Mean}_{\mathbf{x}}(B) = \alpha\big[\mathrm{Mean}_{\mathbf{x}}(\mathrm{Pre}_{B'_+}(i)) - \mathrm{Mean}_{\mathbf{x}}(B'_+)\big]. \tag{20}$$

Hence, for any $i \in B'_+$

$$\begin{aligned}
\mathrm{Mean}_{\mathbf{x}_\varepsilon}(\mathrm{Pre}_{B'_+}(i)) - \mathrm{Mean}_{\mathbf{x}_\varepsilon}(B'_+) &= \mathrm{Mean}_{\mathbf{x}}(\mathrm{Pre}_{B'_+}(i)) + [\![i \geq k]\!] \frac{\varepsilon}{i - i_1} - \mathrm{Mean}_{\mathbf{x}}(B'_+) - \frac{\varepsilon}{|B'_+|} \\
&= \mathrm{Mean}_{\mathbf{x}}(\mathrm{Pre}_{B'_+}(i)) - \mathrm{Mean}_{\mathbf{x}}(B'_+) + \varepsilon \big[\frac{[\![i \geq k]\!]}{i - i_1} - \frac{1}{m} \big] \\
&= \frac{1}{\alpha} \underbrace{\big[\mathrm{Mean}_{\mathbf{x}}(\mathrm{Pre}_B(i)) - \mathrm{Mean}_{\mathbf{x}}(B) \big]}_{D \geq 0} + \varepsilon \underbrace{\big[\frac{[\![i \geq k]\!]}{i - i_1} - \frac{1}{m} \big]}_{F},
\end{aligned}$$

where in the last equality we used eq. (20). What remains is to show that the above quantity is always positive except at the end-point. First, if $i \geq k$, then the F is strictly positive for all points except for the end-point $i = i_1 + m$ in which case it evaluates to zero. Now, let $i < k$. Then, because $i > l(k)$ it can-not be a breakpoint, so that $D > 0$. Then, for all ε smaller than some sufficiently small δ the quantity F will also be positive, which implies dual feasibility for B'_+ and completes the proof. $\square$

Definition 5. *We define $\Sigma(\mathbf{z})$ as the set of permutations that sort $\mathbf{z}$ in descending order, or formally stated*

$$\Sigma(\mathbf{z}) = \{\sigma \in \mathit{Sym}(V) \mid z_{\sigma(1)} \geq z_{\sigma(2)} \geq \ldots \geq z_{\sigma(n)}\}.$$

Proof of theorem 2. We will show that the j-th column of the approximate Jacobian, equal to $\Lambda(\mathbf{y}_{\boldsymbol{\theta}})\partial_{\theta_j}\mathbf{f}_{\boldsymbol{\theta}}(\sigma)$ does not depend on the choice of σ. First, note that the $\sigma(i)$-th coordinate of $\partial_{\theta_j}\mathbf{f}_{\boldsymbol{\theta}}(\sigma)$ is equal to

$$\begin{aligned}
[\partial_{\theta_j} f_{\boldsymbol{\theta}}(\sigma)]_{\sigma(i)} &= \partial_{\theta_j} F_{\boldsymbol{\theta}}(\{\sigma(i)\} \mid \{\sigma(1), \sigma(2), \ldots, \sigma(i-1)\}) \\
&= \partial_{\theta_j} F_{\boldsymbol{\theta}}(\{\sigma(1), \sigma(2), \ldots, \sigma(i)\}) - \partial_{\theta_j} F_{\boldsymbol{\theta}}(\{\sigma(1), \sigma(2), \ldots, \sigma(i-1)\}).
\end{aligned} \tag{21}$$

Let B be the block where i belongs to, i.e. $B = \{i' \mid [\mathbf{y}_{\boldsymbol{\theta}}]_{i'} = [\mathbf{y}_{\boldsymbol{\theta}}]_i\}$. Because we chose a partition σ that sorts $\mathbf{y}_{\boldsymbol{\theta}}$, all elements in B must have their elements placed consecutively under σ, i.e. it there must exist some i_1 so that $B = \{\sigma(i_1 + 1), \sigma(i_2 + 2), \ldots, \sigma(i_1 + |B|)\}$. Then

$$\begin{aligned}
&\mathrm{Mean}_{\partial_{\theta_j}\mathbf{f}_{\boldsymbol{\theta}}(\sigma)}(B) \\
&= \frac{1}{|B|} \sum_{i=i_1+1}^{i_1+|B|} \partial_{\theta_j} F_{\boldsymbol{\theta}}(\{\sigma(i)\} \mid \{\sigma(1), \ldots, \sigma(i_1)\}) \\
&= \frac{1}{|B|} \big[\sum_{i=i_1+1}^{i_1+|B|} \partial_{\theta_j} F_{\boldsymbol{\theta}}(\{\sigma(1), \sigma(2), \ldots, \sigma(i)\}) - \partial_{\theta_j} F_{\boldsymbol{\theta}}(\{\sigma(1), \sigma(2), \ldots, \sigma(i-1)\}) \big] \\
&= \frac{\partial_{\theta_j}}{|B|} F_{\boldsymbol{\theta}}(\{\sigma(1), \ldots, \sigma(i_1 + |B|)\} \mid \{\sigma(1), \ldots, \sigma(i_1)\}) \\
&= \frac{\partial_{\theta_j}}{|B|} F_{\boldsymbol{\theta}}(B \mid \{i' \mid [\mathbf{y}_{\boldsymbol{\theta}}]_{i'} < [\mathbf{y}_{\boldsymbol{\theta}}]_i\}),
\end{aligned}$$

where the third equality follows from the telescoping of the summands. $\square$

Proof of theorem 3. From theorem 1 we can see that the left and right derivatives will agree if and only if each block B_j has no of the chosen order cone. We will refer to such blocks as being *unbreakable*. The first condition then follows from the following lemma.

Lemma 7. *Block B_j is unbreakable if and only if there is no $H \notin \{\emptyset, B_j\}$ satisfying*

$$F_j(H)/|H| = F_j(B_j)/|B_j|.$$

Moreover, this condition is equivalent to $\arg\min_{H \in B_j} \big[F_j(H) - F_j(V)|H|/|B_j| \big] = \{\emptyset, B_j\}$.

Proof. Let w.l.o.g. $B_j = \{1, 2, \ldots, k\}$ and let $\sigma \colon \{1, 2, \ldots, k\} \to \{1, 2, \ldots, k\}$ be the ordering the chosen permutation makes on B_j. Then, at the i-th position inside B_j, the linearization has

$$[\mathbf{f}(\sigma)]_i = F_j(\{\sigma(i)\} \mid \{\sigma(1), \sigma(2), \ldots, \sigma(i-1)\}).$$

Then, due to the telescoping sums we have

$$\text{Mean}_{\mathbf{f}(\sigma)}(\text{Pre}_i(B_j)) = F(\{\sigma(1), \sigma(2), \ldots, \sigma(i)\})/i,$$

with the special case

$$\text{Mean}_{\mathbf{f}(\sigma)}(\text{Pre}_k(B_j)) = F(B_j)/k = F(B)/|B_j|.$$

Now, by definition, B_j is unbreakable if and only if the above two quantities are never equal for $i < k$ for any possible permutation σ, which is exactly what we had to show. □

The second condition will follow as a corollary of the following well-known result characterizing the min-norm point.

Lemma 8. *For each B_j we have that $\mathbf{y}'_{B_j} = F_j(B_j)/|B_j|$.*

Proof. From [1][Theorem 9.1] we know that $y'(A_j) = F_{\boldsymbol{\theta}'}(A_j)$ for each $j \in \{1, \ldots, k\}$. Subtracting the equalities for j and $j-1$ we have

$$y(B_j) = F(A_j) - F(A_{j-1}) = F_j(B_j).$$

Now, the result follows because we have assumed that all elements in B_j are equal. □

Corollary 2. *Block B_j is unbreakable iff $\mathcal{D}_{F_j}(\mathbf{y}'_{B_j}) = \{\emptyset, B_j\}$.*

Proof. Assume there exists some other set $H \subsetneq B_j$ that is tight. This implies that $\mathbf{y}'_{B_j}(H) = F_j(H)$. Then, lemma 8 implies that

$$F_j(B_j)|H|/|B_j| = F_j(H),$$

which is exactly equal to the first condition. □

The fact that the the base polytope has any point in the relative interior being equivalent to inseparability is well-known, see e.g. [2, Prop. 4.6] or [1, Thm. 3.36]. □

Proof of theorem 4. Remember that the min-norm point is defined as

$$\arg\min_{\mathbf{y} \in \mathbb{R}^n} \frac{1}{2}\|\mathbf{y}\|^2$$

subject to $y(A) \le \sum_{j=1}^d \theta_j F_j(A)$ for all $A \subsetneq V$ and $y(V) = \sum_{j=1}^d \theta_j F_j(V)$. Introducing multipliers λ_A for the inequality constraints and μ for the equalities we arrive at the Lagrangian

$$\mathcal{L}(\mathbf{y}, \boldsymbol{\lambda}, \mu) = \frac{1}{2}\|\mathbf{y}\|^2 + \sum_{A \subsetneq V} \lambda_A (y(A) - F_{\boldsymbol{\theta}}(A)) + \mu(y(V) - F(V)).$$

Then, setting $\partial_{y_i}\mathcal{L}$ to zero yields the condition

$$\mu + \sum_{A \ni i} \lambda_A = -y_i. \tag{22}$$

Moreover, due to the complementary slackness condition can have non-zero multipliers only for the tight constraints. As $\mathbf{y}'$ is the min-norm point for $F_{\boldsymbol{\theta}'}$, we know that the sets $A_1, A_2, \ldots, A_k = V$ are all tight (see lemma 8). Remember that $\mathbf{y}'$ took values $\gamma_1 < \gamma_2 < \cdots < \gamma_k$ on sets $B_1, B_2, \ldots, B_k$ respectively. We suggest the following family of multipliers

$$\mu = \gamma_k = -\max_i y_i$$

$$\lambda_A = \begin{cases} \gamma_{j+1} - \gamma_j \ge 0 & \text{if for } A = A_j \text{ for } j \in \{1, 2, \ldots, k-1\}, \text{ or} \\ 0 & \text{otherwise.} \end{cases}$$

They indeed satisfy the condition in eq. (22) because for every $i \in B_j$ we have that

$$\mu + \sum_{A \ni i} \lambda_A = \mu + \sum_{j'=j}^{k-1} \lambda_{A_{j'}} = \gamma_k - \sum_{j'=j}^{k-1} \gamma_{j'+1} - \gamma_{j'} = -\gamma_{j'} = -y_i.$$

The fact that these multipliers are unique will easily follow if we show that the conditions imply that $A_1, A_2, \dots, A_k$ are the only tight sets. Then, it can be easily seen that the only multipliers satisfying eq. (22) are those we have just suggested, by doing an induction on j starting from $j = k$ (corresponding to ν) and going backwards. For the sake of contradiction, assume that the conditions hold and that there exists some H different from the members of the chain that is tight, i.e. $y'(H) = F_{\theta'}(H)$ and $A_{j-1} \subsetneq H \subsetneq A_j$. Then, if we define $H' = H \setminus A_{j-1} \notin \{\emptyset, B_j\}$, we have that

$$\begin{aligned} F_j(H') &= F_{\boldsymbol{\theta}'}(H' \cup A_{j-1}) - F_{\boldsymbol{\theta}'}(A_{j-1}) \\ &= F_{\boldsymbol{\theta}'}(H) - F_{\boldsymbol{\theta}'}(A_j) \\ &= y'(H) - F_{\boldsymbol{\theta}'}(A_j) \\ &= y'(H') - |H'| F_{\boldsymbol{\theta}'}(B_j)/|B_j|, \end{aligned}$$

where the last step follows from lemma 8. However, this means that H' violates the assumption condition for F_j, which is a contradiction to our assumption.

Create the matrix $G = \{0,1\}^{(2^n-1)\times n}$ that has the indicator vectors of the inequality constraints as its rows. Similarly, create the vector $\mathbf{h} \in \mathbb{R}^{2^n-1}$ with elements $F(A)$, and define $b = F(V)$. Let us first compute $\partial \mathbf{y}/\partial b$, for which we have to solve the following system

$$\begin{aligned} d\mathbf{y} + G^T d\boldsymbol{\lambda} + \mathbf{1} d\mu &= \mathbf{0} \\ \mathtt{diag}(\boldsymbol{\lambda}^*) G d\mathbf{y} + \mathtt{diag}(G\mathbf{y} - \mathbf{h}) d\boldsymbol{\lambda} &= \mathbf{0} \\ \mathbf{1}^T d\mathbf{y} &= 1 \end{aligned}$$

We suggest the solution $d\mathbf{y} = \mathbf{1}_{B_k}/|B_k|, d\mu = 1/|B_k|$ and $d\boldsymbol{\lambda}$ to be zero except coordinate $d\boldsymbol{\lambda}_{A_{k-1}} = -1/|B_k|$, which can be easily seen to satisfy the conditions. Namely, the second condition translates to $\mathbf{1}_{A_j}^T d\mathbf{y} = 0$ for $j = 1, 2, \dots, k-1$, which is true at $\mathbf{y}$ is non-zero only on B_k. The third condition is obvious, while the first one reads

$$d\mathbf{y}_i = -\sum_{A \ni i} d\boldsymbol{\lambda}_A + d\mu = -1/|B_k| [\![i \in A_{k-1}]\!] + 1/|B_k| = 1/|B_k| [\![i \notin A_{k-1}]\!] = 1/|B_k| [\![i \in B_k]\!].$$

Now, let us compute $\partial \mathbf{y}/\partial \mathbf{h}$, for which we have to solve the following system

$$\begin{aligned} d\mathbf{y} + G^T d\boldsymbol{\lambda} + \mathbf{1} d\mu &= \mathbf{0} \\ \mathtt{diag}(\boldsymbol{\lambda}) G d\mathbf{y} + \mathtt{diag}(G\mathbf{y} - \mathbf{h}) d\boldsymbol{\lambda} &= \mathtt{diag}(\boldsymbol{\lambda}) \\ \mathbf{1}^T d\mathbf{y} &= 0 \end{aligned}$$

As Ansatz we suggest $d\mu = 0$ and only the following to non-zero

$$d\mathbf{y}_{i,A} = \begin{cases} +1/|B_j| & \text{if } i \in B_j \text{ and } A = A_j \text{ for } j < k, \text{ or} \\ -1/|B_{j+1}| & \text{if } i \in B_{j+1} \text{ and } A = A_j \text{ for } j < k, \text{ or} \\ 0 & \text{otherwise,} \end{cases}$$

and

$$d\boldsymbol{\lambda}_{A',A} = \begin{cases} -1/|B_j| & \text{if } A' = A = A_j \text{ and } j < k, \text{ or} \\ +1/|B_{j+1}| + 1/|B_j| & \text{if } A' = A_j \text{ and } A = A_{j+1}, \text{ if } j < k, \text{ or} \\ 0 & \text{otherwise.} \end{cases}$$

Note that the third condition is immediately satisfied for $d\mathbf{y}$, as every column sums to zero. The third condition for those rows of $d\boldsymbol{\lambda}$ corresponding to $A_1, A_2, \dots, A_{k-1}$ reads

$$\sum_{i \in A_j} d\mathbf{y}_{A',A_j} = [\![A = A_j]\!],$$

which is satisfied because the A_j-th column of $d\mathbf{y}$ is $+1/|B_j|$ on B_j and we do not sum over its other elements which are in B_{j+1}. Finally, the first condition is non-vacuous only for the non-zero columns $A_1, A_2, \dots, A_{k-1}$, in which case it also holds because we have

$$\begin{aligned} d\mathbf{y}_{i,A_j} &= -\sum_{A \ni i} d\boldsymbol{\lambda}_{A,A_j} \\ &= [\![i \in A_j]\!]/|B_j| - (1/|B_j| + 1/|B_{j+1}|)[\![i \in A_{j+1}]\!] \\ &= [\![i \in B_j]\!]/|B_j| - [\![i \in B_{j+1}]\!]/|B_{j+1}|. \end{aligned}$$

Then, if these were the actual true Jacobians, for any parameter θ_j and output $i \in B_j$ we have

$$\begin{aligned}\frac{\partial y_i}{\partial_{\theta_j}} &= \sum_{A \subseteq V} \frac{\partial y_i}{\partial F_\theta(A)} \times \frac{\partial F_\theta(A)}{\theta_j} \\ &= \sum_{j=1}^{k-1} \frac{\partial y_i}{\partial F_\theta(A_j)} \times \frac{\partial F_\theta(A_j)}{\theta_j} \\ &= \underbrace{\frac{\partial y_i}{\partial F_\theta(A_{j-1})}}_{-1/|B_j|} \times \frac{\partial F_\theta(A_j)}{\theta_j} + \underbrace{\frac{\partial y_i}{\partial F_\theta(A_j)}}_{1/|B_j|} \times \frac{\partial F_\theta(A_j)}{\theta_j} = \frac{\partial_{\theta_j}}{|B_j|} F_\theta(B_j).\end{aligned}$$

□

Proof of corollary 1. If the lattices are trivial problem (17) reduces to

$$\min_{\mathbf{y}_k\,:\, \mathbf{y}_k^T \mathbf{1} = G_j(B_j)} \|\mathbf{y}_j\|^2,$$

which has as a solution the constant vector $\mathbf{y}_k^*$ with coordinates all equal to

$$G_j(B_j)/|B_j| = F_{\mathbf{v}}(B_j \mid A_{j-1})/|B_j|.$$

□

Proof of theorem 5. First, because the function $\boldsymbol{\theta} \to \mathbf{y}_{\boldsymbol{\theta}}$ is continuous under the assumptions ([29, Lem. 2.1]), we know that in a neighbourhood of $\boldsymbol{\theta}'$ no inactive constraints can activate. Hence, we can focus on the active constraints only.

Lemma 9. *The four assumptions made in [29] hold whenever $F_{\boldsymbol{\theta}'}$ is inseparable.*

Proof. We show below why each one holds.

1. We need a neighbourhood $N_{\boldsymbol{\theta}'}$ of $\boldsymbol{\theta}'$ so that for all $\boldsymbol{\theta}'' \in N_{\boldsymbol{\theta}'}$ the base polyhedra are bounded. Obviously holds as $\mathcal{B}(F_{\boldsymbol{\theta}}) = \sum_{j=1}^m w_j \mathcal{B}(F_j)$.
2. We have already discussed the uniqueness of the min-norm point.
3. The Mangasarian-Fromovitz condition is implied by Slater's condition, which in case holds whenever the function is inseparable [1][Thm. 3.36].
4. Clearly follows from the strong convexity of the objective.

□

Let us define the shorthands $\mathcal{D}' = \mathcal{D}_{F_{\boldsymbol{\theta}'}}(\mathbf{y}')$ and $\Sigma(\mathbf{v}) = \mathcal{B}(F_{\mathbf{v}} \mid \mathcal{D}')$. It is easy to see that our definition of $\Sigma(\mathbf{v})$ agrees with that in [29][(2.11)] because the constraints are linear in both $\mathbf{y}$ and $\boldsymbol{\theta}$, and there is no term containing both $\mathbf{y}$ and $\boldsymbol{\theta}$. Namely, each constraint is of the form

$$g_A(\mathbf{y}, \boldsymbol{\theta}) = \mathbf{y}^T \mathbf{1}_A - \sum_{j=1}^m \theta_j F_j(A),$$

we have its linear approximation

$$\alpha_A(\mathbf{u}, \mathbf{v}) = \mathbf{u}^T \nabla_{\mathbf{y}} g_A(\mathbf{y}', \boldsymbol{\theta}') + \mathbf{v}^T \nabla_{\boldsymbol{\theta}} g_A(\mathbf{y}', \boldsymbol{\theta}') = \mathbf{u}^T \mathbf{1}_A - \sum_{j=1}^m v_j F_j(A) = u(A) - F_{\mathbf{v}}(A).$$

Then, the equivalence is clear given the definition [29][(2.11)]

$$\Sigma(\mathbf{v}) = \{\mathbf{u} \in \mathbb{R}^n \mid \alpha_A(\mathbf{u}, \mathbf{v}) \le 0 \;\text{ for all } A \in \mathcal{D}'\} \cap \{\mathbf{u} \in \mathbb{R}^n \mid \alpha_V(\mathbf{u}, \mathbf{v}) = 0\}.$$

Lemma 10. *A point* $\mathbf{u} \in \Sigma(\mathbf{v})$ *belongs to* $\overline{\Sigma}(\mathbf{v}) = \arg\min_{\mathbf{u}\in\Sigma(\mathbf{v})} \mathbf{u}^T\mathbf{y}'$ *iff it satisfies*

$$u(A_j) = F_{\mathbf{v}}(A_j) \text{ for } j \in \{1, 2, \ldots, k\}.$$

Proof. First, note that $\mathbf{y}_{\boldsymbol{\theta}}^T \colon V \to \mathbb{R}$ is monotone non-decreasing on $\mathcal{D}'$ because for any $i \in B_i$ we have that $\mathrm{dep}(\mathbf{y}', i) \subseteq A_i$ [1][Theorem 9.1 (iii)], and the lattice we consider is exactly the one generated from the tight sets of the min-norm point. Then, the result follows from [1][Theorem 3.15]. □

Now, because the Lagrangian

$$\mathcal{L}(\mathbf{y}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \nu) = \frac{1}{2}\|\mathbf{y}\|^2 + \sum_{A\in\mathcal{D}'} \lambda_A(y(A) - \sum_{j=1}^{d} \theta_j F_j(A)) + \nu(y(V) + \sum_{j=1}^{d} \theta_j F_j(V))$$

has no terms containing both $\mathbf{y}$ and $\boldsymbol{\theta}$ we have that (defined as (3.3) in [29])

$$\xi_{\boldsymbol{\lambda}}(\mathbf{u}, \mathbf{v}) = \frac{1}{2}\mathbf{u}^T \underbrace{\nabla^2_{\mathbf{yy}}\mathcal{L}}_{I}\, \mathbf{u} + \mathbf{u}^T \underbrace{\nabla^2_{\mathbf{y}\boldsymbol{\theta}}\mathcal{L}}_{0}\, \mathbf{v} + \frac{1}{2}\mathbf{v}^T \underbrace{\nabla^2_{\mathbf{yy}}\mathcal{L}}_{0}\, \mathbf{v} = \frac{1}{2}\|\mathbf{u}\|^2,$$

so that the function is *independent* of its second argument. Then, $\zeta_{\mathbf{v},\mathbf{h}}(\mathbf{u})$ (defined as (4.10) in [29]) reduces to

$$\zeta_{\mathbf{v},\mathbf{h}}(\mathbf{u}) = \frac{1}{2}\|\mathbf{u}\|^2 + z(\mathbf{v}, \mathbf{h}),$$

for some function $z(\mathbf{v}, \mathbf{h})$ independent of $\mathbf{u}$. Then, note that

$$\operatorname*{arg\,min}_{\mathbf{u}\in\overline{\Sigma}(\mathbf{v})} \zeta_{\mathbf{v},\mathbf{h}}(\mathbf{u}) = \operatorname*{arg\,min}_{\mathbf{u}\in\overline{\Sigma}(\mathbf{v})} z(\mathbf{v}, \mathbf{h}) + \frac{1}{2}\|\mathbf{u}\|^2 = \operatorname*{arg\,min}_{\mathbf{u}\in\overline{\Sigma}(\mathbf{v})} \frac{1}{2}\|\mathbf{u}\|^2,$$

hence the optimum does not depend on the choice of $\mathbf{h}$. Then, the claim follows from [29][Theorem 5.1]. □

Proof of lemma 6. We will need the notion of a poset corresponding to $\mathcal{D}'$, which can be found in [1]. If we use this terminology, the sets H_i^* correspond to the union of i and all its ancestors in the poset. We will furthermore define B_1 to be the union of the ancestors and their children, B_2 the descendants of i and B_3 the ideal of all elements in $B_j \setminus B_2$. First notice that the suggested solution is feasible and that B_1, B_2 and B_3 are all in $\mathcal{D}'_j$. We will now prove optimality by showing that it is optimal if we only consider the active set corresponding to B_1, B_2 and B_3. The corresponding Lagrangian is

$$\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{u}\|^2 + \lambda_1(u(B_1) - 1) + \lambda_2(u(B_2) - 0) + \lambda_3(u(B_3) - 0) + \eta(\mathbf{u}^T\mathbf{1} - 1).$$

Setting the derivative wrt u_k for $k \in B_1$ to zero yields the condition

$$u_k = -\lambda_1 - \lambda_2[\![k \in B_2]\!] - \eta.$$

For any $k \in H_i^* = B_1 \setminus B_2$ we have to satisfy $-\lambda_1 - \eta = 1/|H_i^*|$, so as Ansatz take $\lambda_1 = 1/|A_1|$ and $\eta = -2/|A_1|$. Then, to satisfy the condition for any other $k \in B_1$ we can use $\lambda_2 = -\lambda_1 - \eta = 1/|H_i^*|$, which is also positive. Finally, for any $k \notin B_1$ the condition is satisfied by using the multiplier $\lambda_3 = 2/|A_i|$. □

C Experiments

Architectures. We used the following architectures in our experiments. For an exact specification of the modules please consult the documentation of `pytorch`.

```
# The network that predicts the per-pixel scores.
nn.Sequential(nn.Conv2d(3,  32, 3, padding=1),
              nn.ReLU(),
              nn.Conv2d(32, 64, 3, padding=1),
              nn.ReLU(),
              nn.Conv2d(64, 1,  3, padding=1))

# The networks that predict the log of the weights.
nn.Sequential(nn.Conv2d(3,  32, 3, padding=1),
              nn.ReLU(),
              nn.Conv2d(32, 1, 3, padding=1))
```

Optimization. We have obtained the best results from the following optimizers: Adam with learning rate 10^{-3} for CNN+GC, and Adagrad with learning rate 10^{-2} for CNN. We have trained for a total of every 100 epochs, evaluated on the validation set after each epoch and chose the model with the best validation score. We have used the default parameter initialization from `PyTorch`.

D Two dimensional total variation code

```
from prox_tv import tv1w_2d  # The total variation solver.
import numpy as np
import torch
from torch.autograd import Function


class TotalVariation2d(Function):
    """A two dimensional total varition function.

    Specifically, given as input (‘x‘, ‘w_r‘, ‘w_c‘), the output is

     argmin_z 0.5 |x-z|^2 + \sum_{i, j} w_r_{i,j} |x_{i, j} - x_{i, j + 1}|
                          + \sum_{i, j} w_c_{i,j} |x_{i, j} - x_{i + 1, j}|."""

    def forward(self, x, weights_row, weights_col):
        """Solves the TV problem and returns the solution.

        Arguments
        ---------
            x : [m, n] matrix holding the input signal
            weights_row : [m, n - 1] matrix holding the horizontal weights
            weights_col : [m - 1, n] matrix holding the vertical weights

        Returns
        -------
            The TV solution, matrix of size [m, n]"""
        opt = tv1w_2d(- x.numpy(), weights_col.numpy(), weights_row.numpy())
        opt = torch.Tensor(opt)
        self.save_for_backward(opt)
        return opt

    def backward(self, grad_output):
        opt, = self.saved_tensors
        grad_x = torch.zeros(opt.size())  # We always compute this derivative.

        values = np.unique(opt.numpy())  # One group for each unique value.
        groups = [(opt.numpy() == val).reshape(opt.size()) for val in values]
        for group in groups:
            grad_x.numpy()[group] = - np.mean(grad_output.numpy()[group])

        diffs_row = torch.sign(opt[:, :-1] - opt[:, 1:])
        grad_weights_row = diffs_row * (grad_x[:, :-1] - grad_x[:, 1:])

        diffs_col = torch.sign(opt[:-1, :] - opt[1:, :])
        grad_weights_col = diffs_col * (grad_x[:-1, :] - grad_x[1:, :])

        return grad_x, grad_weights_row, grad_weights_col
```

Figure 3: A `pytorch` function doing the forward and backward passes.