SparseMAP: Differentiable Sparse Structured Inference

Vlad Niculae [1]   André F. T. Martins [2]   Mathieu Blondel [3]   Claire Cardie [1]

Abstract

Structured prediction requires searching over a combinatorial number of structures. To tackle it, we introduce SparseMAP: a new method for **sparse structured inference**, and its natural loss function. SparseMAP automatically selects only a few global structures: it is situated between MAP inference, which picks a single structure, and marginal inference, which assigns nonzero probability to all structures, including implausible ones. SparseMAP can be computed using only calls to a MAP oracle, making it applicable to problems with intractable marginal inference, *e.g.*, linear assignment. Sparsity makes gradient backpropagation efficient regardless of the structure, enabling us to augment deep neural networks with generic and sparse **structured hidden layers**. Experiments in dependency parsing and natural language inference reveal competitive accuracy, improved interpretability, and the ability to capture natural language ambiguities, which is attractive for pipeline systems.

1. Introduction

Structured prediction involves the manipulation of discrete, combinatorial structures, *e.g.*, trees and alignments (Bakır et al., 2007; Smith, 2011; Nowozin et al., 2014). Such structures arise naturally as machine learning outputs, and as intermediate representations in deep pipelines. However, the set of possible structures is typically prohibitively large. As such, inference is a core challenge, often sidestepped by greedy search, factorization assumptions, or continuous relaxations (Belanger & McCallum, 2016).

[1]Cornell University, Ithaca, NY [2]Unbabel & Instituto de Telecomunicações, Lisbon, Portugal [3]NTT Communication Science Laboratories, Kyoto, Japan. Correspondence to: Vlad Niculae <vlad@vene.ro>, André F. T. Martins <andre.martins@unbabel.com>, Mathieu Blondel <mathieu@mblondel.org>, Claire Cardie <cardie@cs.cornell.edu>.
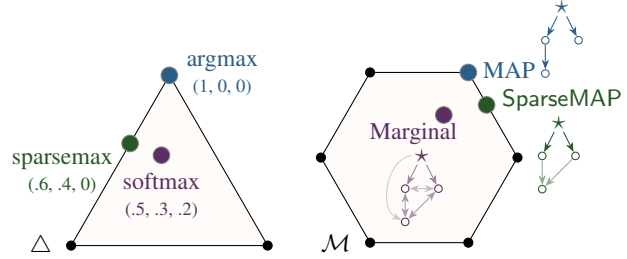
Figure 1. Left: in the unstructured case, softmax and sparsemax can be interpreted as regularized, differentiable arg max approximations; softmax returns dense solutions while sparsemax favors sparse ones. Right: in this work, we extend this view to *structured inference*, which consists of optimizing over a polytope $\mathcal{M}$, the convex hull of all possible structures (depicted: the arborescence polytope, whose vertices are trees). We introduce SparseMAP as a structured extension of sparsemax: it is situated in between MAP inference, which yields a single structure, and marginal inference, which returns a dense combination of structures.

In this paper, we propose an appealing alternative: a new inference strategy, dubbed SparseMAP, which encourages **sparsity** in the structured representations. Namely, we seek solutions explicitly expressed as a combination of a small, enumerable set of global structures. Our framework departs from the two most common inference strategies in structured prediction: **maximum *a posteriori* (MAP) inference**, which returns the highest-scoring structure, and **marginal inference**, which yields a dense probability distribution over structures. Neither of these strategies is fully satisfactory: for latent structure models, marginal inference is appealing, since it can represent uncertainty and, unlike MAP inference, it is continuous and differentiable, hence amenable for use in structured hidden layers in neural networks (Kim et al., 2017). It has, however, several limitations. First, there are useful problems for which MAP is tractable, but marginal inference is not, *e.g.*, linear assignment (Valiant, 1979; Taskar, 2004). Even when marginal inference is available, case-by-case derivation of the backward pass is needed, sometimes producing fairly complicated algorithms, *e.g.*, second-order expectation semirings (Li & Eisner, 2009). Finally, marginal inference is *dense*: it assigns nonzero probabilities to all structures and cannot *completely rule out* irrelevant ones. This can be statistically and computationally wasteful, as well as qualitatively harder to interpret.

In this work, we make the following contributions:

1. We propose SparseMAP: a new framework for **sparse structured inference** (§3.1). The main idea is illustrated in Figure 1. SparseMAP is a twofold generalization: first, as a structured extension of the sparsemax transformation (Martins & Astudillo, 2016); second, as a continuous yet sparse relaxation of MAP inference. MAP yields a single structure and marginal inference yields a dense distribution over *all* structures. In contrast, the SparseMAP solutions are sparse combinations of a small number of often-overlapping structures.

2. We show how to **compute** SparseMAP **effectively**, requiring only a MAP solver as a subroutine (§3.2), by exploiting the problem's sparsity and quadratic curvature. Noticeably, the MAP oracle can be any arbitrary solver, *e.g.*, the Hungarian algorithm for linear assignment, which permits tackling problems for which marginal inference is intractable.

3. We derive expressions for **gradient backpropagation** through SparseMAP inference, which, unlike MAP, is differentiable almost everywhere (§3.3). The backward pass is fully general (applicable to any type of structure), and it is efficient, thanks to the sparsity of the solutions and to reusing quantities computed in the forward pass.

4. We introduce a novel SparseMAP **loss** for structured prediction, placing it into a family of loss functions which generalizes the CRF and structured SVM losses (§4). Inheriting the desirable properties of SparseMAP inference, the SparseMAP loss and its gradients can be computed efficiently, provided access to MAP inference.

Our experiments demonstrate that SparseMAP is useful both for predicting structured outputs, as well as for learning latent structured representations. On *dependency parsing* (§5.1), structured output networks trained with the SparseMAP loss yield more accurate models with sparse, interpretable predictions, adapting to the ambiguity (or lack thereof) of test examples. On *natural language inference* (§5.2), we learn latent structured alignments, obtaining good predictive performance, as well as useful natural visualizations concentrated on a small number of structures.[1]

Notation. Given vectors $\boldsymbol{a} \in \mathbb{R}^m, \boldsymbol{b} \in \mathbb{R}^n$, $[\boldsymbol{a}; \boldsymbol{b}] \in \mathbb{R}^{m+n}$ denotes their concatenation; given matrices $\boldsymbol{A} \in \mathbb{R}^{m \times k}, \boldsymbol{B} \in \mathbb{R}^{n \times k}$, we denote their row-wise stacking as $[\boldsymbol{A}; \boldsymbol{B}] \in \mathbb{R}^{(m+n) \times k}$. We denote the columns of a matrix $\boldsymbol{A}$ by $\boldsymbol{a}_j$; by extension, a slice of columns of $\boldsymbol{A}$ is denoted $\boldsymbol{A}_{\mathcal{I}}$ for a set of indices $\mathcal{I}$. We denote the canonical simplex by $\triangle^d := \{\boldsymbol{y} \in \mathbb{R}^d : \boldsymbol{y} \succeq \boldsymbol{0}, \sum_{i=1}^d y_i = 1\}$, and the indicator function of a predicate p as $\mathbb{I}[p] = \{1 \text{ if } p, \ 0 \text{ otherwise }\}$.

---

[1] General-purpose dynet and pytorch implementations available at https://github.com/vene/sparsemap.

2. Preliminaries

2.1. Regularized Max Operators: Softmax, Sparsemax

As a basis for the more complex structured case, we first consider the simple problem of selecting the largest value in a vector $\boldsymbol{\theta} \in \mathbb{R}^d$. We denote the vector mapping

$$\arg\max(\boldsymbol{\theta}) := \arg\max_{\boldsymbol{y} \in \triangle^d} \boldsymbol{\theta}^\top \boldsymbol{y}.$$

When there are no ties, $\arg\max$ has a unique solution $\boldsymbol{e}_i$ peaking at the index i of the highest value of $\boldsymbol{\theta}$. When there are ties, $\arg\max$ is set-valued. Even assuming no ties, $\arg\max$ is piecewise constant, and thus is ill-suited for direct use within neural networks, *e.g.*, in an attention mechanism. Instead, it is common to use softmax, a continuous and differentiable approximation to $\arg\max$, which can be seen as an entropy-regularized $\arg\max$

$$\mathsf{softmax}(\boldsymbol{\theta}) := \arg\max_{\boldsymbol{y} \in \triangle^d} \boldsymbol{\theta}^\top \boldsymbol{y} + H(\boldsymbol{y}) = \frac{\exp \boldsymbol{\theta}}{\sum_{i=1}^d \exp \theta_i} \quad (1)$$

where $H(\boldsymbol{y}) = -\sum_i y_i \ln y_i$, *i.e.* the negative Shannon entropy. Since $\exp \cdot > 0$ strictly, softmax outputs are dense.

By replacing the entropic penalty with a squared ℓ_2 norm, Martins & Astudillo (2016) introduced a sparse alternative to softmax, called sparsemax, given by

$$\begin{aligned} \mathsf{sparsemax}(\boldsymbol{\theta}) &:= \arg\max_{\boldsymbol{y} \in \triangle^d} \boldsymbol{\theta}^\top \boldsymbol{y} - \frac{1}{2} \|\boldsymbol{y}\|_2^2 \\ &= \arg\min_{\boldsymbol{y} \in \triangle^d} \|\boldsymbol{y} - \boldsymbol{\theta}\|_2^2. \end{aligned} \quad (2)$$

Both softmax and sparsemax are continuous and differentiable almost everywhere; however, sparsemax encourages sparsity in its outputs. This is because it corresponds to an Euclidean projection onto the simplex, which is likely to hit its boundary as the magnitude of $\boldsymbol{\theta}$ increases. Both mechanisms, as well as variants with different penalties (Niculae & Blondel, 2017), have been successfully used in attention mechanisms, for mapping a score vector $\boldsymbol{\theta}$ to a d-dimensional normalized discrete probability distribution over a small set of choices. The relationship between $\arg\max$, softmax, and sparsemax, illustrated in Figure 1, sits at the foundation of SparseMAP.

2.2. Structured Inference

In structured prediction, the space of possible outputs is typically very large: for instance, all possible labelings of a length-n sequence, spanning trees over n nodes, or one-to-one alignments between two sets. We may still write optimization problems such as $\max_{s=1}^D \theta_s$, but it is impractical to enumerate all of the D possible structures and, in turn, to specify the scores for each structure in $\boldsymbol{\theta}$.

Instead, structured problems are often parametrized through **structured log-potentials** (scores) $\boldsymbol{\theta} := \boldsymbol{A}^\top \boldsymbol{\eta}$, where $\boldsymbol{A} \in \mathbb{R}^{k \times D}$ is a matrix that specifies the structure of the problem, and $\boldsymbol{\eta} \in \mathbb{R}^k$ is lower-dimensional parameter vector, *i.e.*, $k \ll D$. For example, in a **factor graph** (Kschischang et al., 2001) with variables U and factors F, $\boldsymbol{\theta}$ is given by

$$\theta_s := \sum_{i \in U} \eta_{U,i}(s_i) + \sum_{f \in F} \eta_{F,f}(s_f),$$

where $\boldsymbol{\eta}_U$ and $\boldsymbol{\eta}_F$ are unary and higher-order log-potentials, and s_i and s_f are local configurations at variable and factor nodes. This can be written in matrix notation as $\boldsymbol{\theta} = \boldsymbol{M}^\top \boldsymbol{\eta}_U + \boldsymbol{N}^\top \boldsymbol{\eta}_F$ for suitable matrices $\{\boldsymbol{M}, \boldsymbol{N}\}$, fitting the assumption above with $\boldsymbol{A} = [\boldsymbol{M}; \boldsymbol{N}]$ and $\boldsymbol{\eta} = [\boldsymbol{\eta}_U; \boldsymbol{\eta}_F]$.

We can then rewrite the **MAP inference** problem, which seeks the highest-scoring structure, as a k-dimensional problem, by introducing variables $[\boldsymbol{u}; \boldsymbol{v}] \in \mathbb{R}^k$ to denote configurations at variable and factor nodes:[2]

$$
\begin{aligned}
\mathsf{MAP}_{\boldsymbol{A}}(\boldsymbol{\eta}) &:= \underset{\substack{\boldsymbol{u} := \boldsymbol{M}\boldsymbol{y} \\ \boldsymbol{y} \in \triangle^D}}{\arg\max} \quad \boldsymbol{\theta}^\top \boldsymbol{y} \\
&= \underset{\boldsymbol{u}:\, [\boldsymbol{u};\boldsymbol{v}] \in \mathcal{M}_{\boldsymbol{A}}}{\arg\max} \quad \boldsymbol{\eta}_U^\top \boldsymbol{u} + \boldsymbol{\eta}_F^\top \boldsymbol{v},
\end{aligned}
\tag{3}
$$

where $\mathcal{M}_{\boldsymbol{A}} := \{[\boldsymbol{u}; \boldsymbol{v}] : \boldsymbol{u} = \boldsymbol{M}\boldsymbol{y},\ \boldsymbol{v} = \boldsymbol{N}\boldsymbol{y},\ \boldsymbol{y} \in \triangle^D\}$ is the **marginal polytope** (Wainwright & Jordan, 2008), with one vertex for each possible structure (Figure 1). However, as previously said, since it is equivalent to a D-dimensional arg max, MAP is piecewise constant and discontinuous.

Negative entropy regularization over $\boldsymbol{y}$, on the other hand, yields **marginal inference**,

$$
\begin{aligned}
\mathsf{Marginal}_{\boldsymbol{A}}(\boldsymbol{\eta}) &:= \underset{\substack{\boldsymbol{u} := \boldsymbol{M}\boldsymbol{y} \\ \boldsymbol{y} \in \triangle^D}}{\arg\max} \quad \boldsymbol{\theta}^\top \boldsymbol{y} + H(\boldsymbol{y}) \\
&= \underset{\boldsymbol{u}:\, [\boldsymbol{u};\boldsymbol{v}] \in \mathcal{M}_{\boldsymbol{A}}}{\arg\max} \quad \boldsymbol{\eta}_U^\top \boldsymbol{u} + \boldsymbol{\eta}_F^\top \boldsymbol{v} + H_{\boldsymbol{A}}(\boldsymbol{u}, \boldsymbol{v}).
\end{aligned}
\tag{4}
$$

Marginal inference is differentiable, but may be more difficult to compute; the entropy $H_{\boldsymbol{A}}(\boldsymbol{u}, \boldsymbol{v}) = H(\boldsymbol{y})$ itself lacks a closed form (Wainwright & Jordan, 2008, §4.1.2). Gradient backpropagation is available only to specialized problem instances, *e.g.* those solvable by dynamic programming (Li & Eisner, 2009). The entropic term regularizes $\boldsymbol{y}$ toward more uniform distributions, resulting in strictly dense solutions, just like in the case of softmax (Equation 1).

Interesting types of structures, which we use in the experiments described in Section 5, include the following.

---

[2] We use the notation $\arg\max_{\boldsymbol{u}:\, [\boldsymbol{u};\boldsymbol{v}] \in \mathcal{M}}$ to convey that the maximization is over both $\boldsymbol{u}$ and $\boldsymbol{v}$, but only $\boldsymbol{u}$ is returned. Separating the variables as $[\boldsymbol{u}; \boldsymbol{v}]$ loses no generality and allows us to isolate the unary posteriors $\boldsymbol{u}$ as the return value of interest.

Sequence tagging. Consider a sequence of n items, each assigned one out of a possible m tags. In this case, a global structure s is a joint assignment of tags $(t_1, \cdots, t_n)$. The matrix $\boldsymbol{M}$ is nm-by-m^n–dimensional, with columns $\boldsymbol{m}_s \in \{0,1\}^{nm} := [\boldsymbol{e}_{t_1}, ..., \boldsymbol{e}_{t_n}]$ indicating which tag is assigned to each variable in the global structure s. $\boldsymbol{N}$ is nm^2-by-m^n–dimensional, with $\boldsymbol{n}_s$ encoding the transitions between consecutive tags, *i.e.*, $\boldsymbol{n}_s(i, a, b) := \mathbb{I}[t_{i-1} = a\ \&\ t_i = b]$. The Viterbi algorithm provides MAP inference and forward-backward provides marginal inference (Rabiner, 1989).

Non-projective dependency parsing. Consider a sentence of length n. Here, a structure s is a dependency tree: a rooted spanning tree over the n^2 possible arcs (for example, the arcs above the sentences in Figure 3). Each column $\boldsymbol{m}_s \in \{0,1\}^{n^2}$ encodes a tree by assigning a 1 to its arcs. $\boldsymbol{N}$ is empty, $\mathcal{M}_{\boldsymbol{A}}$ is known as the *arborescence polytope* (Martins et al., 2009). MAP inference may be performed by *maximal arborescence* algorithms (Chu & Liu, 1965; Edmonds, 1967; McDonald et al., 2005), and the Matrix-Tree theorem (Kirchhoff, 1847) provides a way to perform marginal inference (Koo et al., 2007; Smith & Smith, 2007).

Linear assignment. Consider a one-to-one matching (linear assignment) between two sets of n nodes. A global structure s is a n-permutation, and a column $\boldsymbol{m}_s \in \{0,1\}^{n^2}$ can be seen as a flattening of the corresponding permutation matrix. Again, $\boldsymbol{N}$ is empty. $\mathcal{M}_{\boldsymbol{A}}$ is the Birkhoff polytope (Birkhoff, 1946), and MAP inference can be performed by, *e.g.*, the Hungarian algorithm (Kuhn, 1955) or the Jonker-Volgenant algorithm (Jonker & Volgenant, 1987). Noticeably, marginal inference is known to be #P-complete (Valiant, 1979; Taskar, 2004, Section 3.5). This makes it an open problem how to use matchings as latent variables.

3. SparseMAP

Armed with the parallel between structured inference and regularized max operators described in §2, we are now ready to introduce SparseMAP, a novel inference optimization problem which returns sparse solutions.

3.1. Definition

We introduce SparseMAP by regularizing the MAP inference problem in Equation 3 with a squared ℓ_2 penalty on the returned posteriors, *i.e.*, $\frac{1}{2}\|\boldsymbol{u}\|_2^2$. Denoting, as above, $\boldsymbol{\theta} := \boldsymbol{A}^\top \boldsymbol{\eta}$, the result is a quadratic optimization problem,

$$
\begin{aligned}
\mathsf{SparseMAP}_{\boldsymbol{A}}(\boldsymbol{\eta}) &:= \underset{\substack{\boldsymbol{u} := \boldsymbol{M}\boldsymbol{y} \\ \boldsymbol{y} \in \triangle^D}}{\arg\max} \quad \boldsymbol{\theta}^\top \boldsymbol{y} - \frac{1}{2}\|\boldsymbol{M}\boldsymbol{y}\|_2^2 \\
&= \underset{\boldsymbol{u}:\, [\boldsymbol{u};\boldsymbol{v}] \in \mathcal{M}_{\boldsymbol{A}}}{\arg\max} \quad \boldsymbol{\eta}_U^\top \boldsymbol{u} + \boldsymbol{\eta}_F^\top \boldsymbol{v} - \frac{1}{2}\|\boldsymbol{u}\|_2^2.
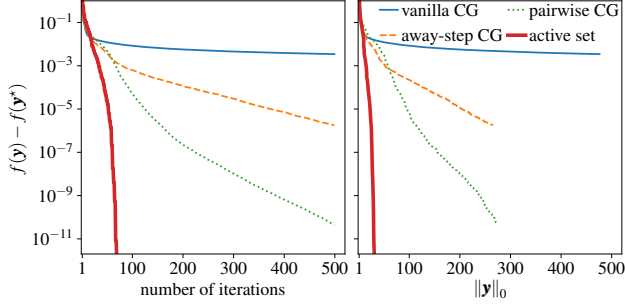\end{aligned}
\tag{5}
$$

Figure 2. Comparison of solvers on the SparseMAP optimization problem for a tree factor with 20 nodes. The active set solver converges much faster and to a much sparser solution.

The quadratic penalty replaces the entropic penalty from marginal inference (Equation 4), which pushes the solutions to the strict interior of the marginal polytope. In consequence, SparseMAP favors sparse solutions from the faces of the marginal polytope $\mathcal{M}_A$, as illustrated in Figure 1. For the structured prediction problems mentioned in Section 2.2, SparseMAP would be able to return, for example, a sparse combination of sequence labelings, parse trees, or matchings. Moreover, the strongly convex regularization on $\boldsymbol{u}$ ensures that SparseMAP has a unique solution and is differentiable almost everywhere, as we will see.

3.2. Solving SparseMAP

We now tackle the optimization problem in Equation 5. Although SparseMAP is a QP over a polytope, even describing it in standard form is infeasible, since enumerating the exponentially-large set of vertices is infeasible. This prevents direct application of, *e.g.*, the generic differentiable QP solver of Amos & Kolter (2017). We instead focus on SparseMAP solvers that involve a sequence of MAP problems as a subroutine—this makes SparseMAP widely applicable, given the availability of MAP implementations for various structures. We discuss two such methods, one based on the conditional gradient algorithm and another based on the active set method for quadratic programming. We provide a full description of both methods in Appendix A.

Conditional gradient. One family of such solvers is based on the *conditional gradient* (CG) algorithm (Frank & Wolfe, 1956; Lacoste-Julien & Jaggi, 2015), considered in prior work for solving approximations of the marginal inference problem (Belanger et al., 2013; Krishnan et al., 2015). Each step must solve a linearized subproblem. Denote by f the SparseMAP objective from Equation 5,

$$f(\boldsymbol{u}, \boldsymbol{v}) \coloneqq \boldsymbol{\eta}_U^\top \boldsymbol{u} + \boldsymbol{\eta}_F^\top \boldsymbol{v} - \frac{1}{2} \|\boldsymbol{u}\|_2^2.$$

The gradients of f with respect to the two variables are

$$\nabla_{\boldsymbol{u}} f(\boldsymbol{u}', \boldsymbol{v}') = \boldsymbol{\eta}_U - \boldsymbol{u}', \qquad \nabla_{\boldsymbol{v}} f(\boldsymbol{u}', \boldsymbol{v}') = \boldsymbol{\eta}_V.$$

A linear approximation to f around a point $[\boldsymbol{u}'; \boldsymbol{v}']$ is

$$\hat{f}(\boldsymbol{u}, \boldsymbol{v}) \coloneqq (\nabla_{\boldsymbol{u}} f)^\top \boldsymbol{u} + (\nabla_{\boldsymbol{v}} f)^\top \boldsymbol{v} = (\boldsymbol{\eta}_U - \boldsymbol{u}')^\top \boldsymbol{u} + \boldsymbol{\eta}_F^\top \boldsymbol{v}.$$

Minimizing $\hat{f}$ over $\mathcal{M}$ is exactly MAP inference with adjusted variable scores $\boldsymbol{\eta}_U - \boldsymbol{u}'$. Intuitively, at each step we seek a high-scoring structure while penalizing sharing variables with already-selected structures Vanilla CG simply adds the new structure to the active set at every iteration. The pairwise and away-step variants trade off between the direction toward the new structure, and away from one of the already-selected structures. More sophisticated variants have been proposed (Garber & Meshi, 2016) which can provide sparse solutions when optimizing over a polytope.

Active set method. Importantly, the SparseMAP problem in Equation 5 has quadratic curvature, which the general CG algorithms may not optimally leverage. For this reason, we consider the active set method for constrained QPs: a generalization of Wolfe's min-norm point algorithm (Wolfe, 1976), also used in structured prediction for the quadratic subproblems by Martins et al. (2015). The active set algorithm, at each iteration, updates an estimate of the solution support by adding or removing one constraint to/from the active set; then it solves the Karush–Kuhn–Tucker (KKT) system of a relaxed QP restricted to the current support.

Comparison. Both algorithms enjoy global linear convergence with similar rates (Lacoste-Julien & Jaggi, 2015), but the active set algorithm also exhibits exact finite convergence—this allows it, for instance, to capture the optimal sparsity pattern (Nocedal & Wright, 1999, Ch. 16.4 & 16.5). Vinyes & Obozinski (2017) provide a more in-depth discussion of the connections between the two algorithms. We perform an empirical comparison on a dependency parsing instance with random potentials. Figure 2 shows that active set substantially outperforms all CG variants, both in terms of objective value as well as in the solution sparsity, suggesting that the quadratic curvature makes SparseMAP solvable in very few iterations to high accuracy. We therefore use the active set solver in the remainder of the paper.

3.3. Backpropagating Gradients through SparseMAP

In order to use SparseMAP as a neural network layer trained with backpropagation, one must compute products of the SparseMAP Jacobian with a vector $\boldsymbol{p}$. Computing the Jacobian of an optimization problem is an active research topic known as *argmin differentiation*, and is generally difficult. Fortunately, as we show next, argmin differentiation is always easy and efficient in the case of SparseMAP.

Proposition 1 *Denote a* SparseMAP *solution by* $\boldsymbol{y}^\star$ *and its support by* $\mathcal{I} \coloneqq \{s \ : \ y_s > 0\}$*. Then,* SparseMAP *is*

differentiable almost everywhere with Jacobian

$$\frac{\partial \boldsymbol{u}^{\star}}{\partial \boldsymbol{\eta}} = \boldsymbol{M}\boldsymbol{D}(\mathcal{I})\boldsymbol{A}^{\top}, \text{ where } \boldsymbol{D}(\mathcal{I}) = \boldsymbol{D}(\mathcal{I})^{\top} \text{ given by}$$

$$\boldsymbol{d}(\mathcal{I})_s := \begin{cases} \left(\boldsymbol{I} - \frac{1}{\mathbf{1}^{T}\boldsymbol{Z}\mathbf{1}}\boldsymbol{Z}\mathbf{1}\mathbf{1}^{T}\right)\boldsymbol{z}_s, & s \in \mathcal{I} \\ \mathbf{0} & s \notin \mathcal{I} \end{cases},$$

$$\boldsymbol{Z} := (\boldsymbol{M}_{\mathcal{I}}^{\top}\boldsymbol{M}_{\mathcal{I}})^{-1}.$$

The proof, given in Appendix B, relies on the KKT conditions of the SparseMAP QP. Importantly, because $\boldsymbol{D}(\mathcal{I})$ is zero outside of the support of the solution, computing the Jacobian only requires the columns of $\boldsymbol{M}$ and $\boldsymbol{A}$ corresponding to the structures in the active set. Moreover, when using the active set algorithm discussed in §3.2, the matrix $\boldsymbol{Z}$ is readily available as a byproduct of the forward pass. The backward pass can, therefore, be computed in $\mathcal{O}(k|\mathcal{I}|)$.

Our approach for gradient computation draws its efficiency from the solution sparsity and does not depend on the type of structure considered. This is contrasted with two related lines of research. The first is "unrolling" iterative inference algorithms, for instance belief propagation (Stoyanov et al., 2011; Domke, 2013) and gradient descent (Belanger et al., 2017), where the backward pass complexity scales with the number of iterations. In the second, employed by Kim et al. (2017), when inference can be performed via dynamic programming, backpropagation can be performed using second-order expectation semirings (Li & Eisner, 2009) or more general smoothing (Mensch & Blondel, 2018), in the same time complexity as the forward pass. Moreover, in our approach, neither the forward nor the backward passes involve logarithms, exponentiations or log-domain classes, avoiding the slowdown and stability issues normally incurred.

In the unstructured case, since $\boldsymbol{M} = \boldsymbol{I}$, $\boldsymbol{Z}$ is also an identity matrix, uncovering the sparsemax Jacobian (Martins & Astudillo, 2016). In general, structures are not necessarily orthogonal, but may have degrees of overlap.

4. Structured Fenchel-Young Losses and the SparseMAP Loss

With the efficient algorithms derived above in hand, we switch gears to defining a SparseMAP *loss function*. Structured output prediction models are typically trained by minimizing a *structured loss* measuring the discrepancy between the desired structure (encoded, for instance, as an indicator vector $\boldsymbol{y} = \boldsymbol{e}_s$) and the prediction induced by the log-potentials $\boldsymbol{\eta}$. We provide here a general family of structured prediction losses that will make the newly proposed SparseMAP loss arise as a very natural case. Below, we let $\Omega : \mathbb{R}^D \to \mathbb{R}$ denote a convex penalty function and denote

by $\Omega_{\triangle}$ its restriction to $\triangle^D \subset \mathbb{R}^D$, *i.e.*,

$$\Omega_{\triangle}(\boldsymbol{y}) := \begin{cases} \Omega(\boldsymbol{y}), & \boldsymbol{y} \in \triangle^D; \\ \infty, & \boldsymbol{y} \notin \triangle^D. \end{cases}$$

The Fenchel convex conjugate of $\Omega_{\triangle}$ is

$$\Omega_{\triangle}^{\star}(\boldsymbol{\theta}) := \sup_{\boldsymbol{y} \in \mathbb{R}^D} \boldsymbol{\theta}^{\top}\boldsymbol{y} - \Omega_{\triangle}(\boldsymbol{y}) = \sup_{\boldsymbol{y} \in \triangle^D} \boldsymbol{\theta}^{\top}\boldsymbol{y} - \Omega(\boldsymbol{y}).$$

We next introduce a family of structured prediction losses, named after the corresponding Fenchel-Young duality gap.

Definition 1 (Fenchel-Young losses) *Given a convex penalty function $\Omega : \mathbb{R}^D \to \mathbb{R}$, and a $(k \times D)$-dimensional matrix $\boldsymbol{A} = [\boldsymbol{M}; \boldsymbol{N}]$ encoding the structure of the problem, we define the following family of structured losses:*

$$\ell_{\Omega, \boldsymbol{A}}(\boldsymbol{\eta}, \boldsymbol{y}) := \Omega_{\triangle}^{\star}(\boldsymbol{A}^{\top}\boldsymbol{\eta}) + \Omega_{\triangle}(\boldsymbol{y}) - \boldsymbol{\eta}^{\top}\boldsymbol{A}\boldsymbol{y}. \quad (6)$$

This family, studied in more detail in (Blondel et al., 2018), includes the commonly-used structured losses:

- Structured perceptron (Collins, 2002): $\Omega \equiv 0$;
- Structured SVM (Taskar et al., 2003; Tsochantaridis et al., 2004): $\Omega \equiv \rho(\cdot, \bar{\boldsymbol{y}})$ for a cost function ρ, where $\bar{\boldsymbol{y}}$ is the true output;
- CRF (Lafferty et al., 2001): $\Omega \equiv -H$;
- Margin CRF (Gimpel & Smith, 2010): $\Omega \equiv -H + \rho(\cdot, \bar{\boldsymbol{y}})$.

This leads to a natural way of defining SparseMAP losses, by plugging the following into Equation 6:

- SparseMAP loss: $\Omega(\boldsymbol{y}) = \frac{1}{2}\|\boldsymbol{M}\boldsymbol{y}\|_2^2$,
- Margin SparseMAP: $\Omega(\boldsymbol{y}) = \frac{1}{2}\|\boldsymbol{M}\boldsymbol{y}\|_2^2 + \rho(\boldsymbol{y}, \bar{\boldsymbol{y}})$.

It is well-known that the subgradients of structured perceptron and SVM losses consist of MAP inference, while the CRF loss gradient requires marginal inference. Similarly, the subgradients of the SparseMAP loss can be computed via SparseMAP inference, which in turn only requires MAP. The next proposition states properties of structured Fenchel-Young losses, including a general connection between a loss and its corresponding inference method.

Proposition 2 *Consider a convex Ω and a structured model defined by the matrix $\boldsymbol{A} \in \mathbb{R}^{k \times D}$. Denote the inference objective $f_{\Omega}(\boldsymbol{y}) := \boldsymbol{\eta}^{\top}\boldsymbol{A}\boldsymbol{y} - \Omega(\boldsymbol{y})$, and a solution $\boldsymbol{y}^{\star} := \arg\max_{\boldsymbol{y} \in \triangle^D} f_{\Omega}(\boldsymbol{y})$. Then, the following properties hold:*

1. $\ell_{\Omega, \boldsymbol{A}}(\boldsymbol{\eta}, \boldsymbol{y}) \geq 0$, with equality when $f_{\Omega}(\boldsymbol{y}) = f_{\Omega}(\boldsymbol{y}^{\star})$;

2. $\ell_{\Omega, \boldsymbol{A}}(\boldsymbol{\eta}, \boldsymbol{y})$ is convex, $\partial\ell_{\Omega, \boldsymbol{A}}(\boldsymbol{\eta}, \boldsymbol{y}) \ni \boldsymbol{A}(\boldsymbol{y}^{\star} - \boldsymbol{y})$;

3. $\ell_{t\Omega, \boldsymbol{A}}(\boldsymbol{\eta}, \boldsymbol{y}) = t\ell_{\Omega}(\boldsymbol{\eta}/t, \boldsymbol{y})$ for any $t \in \mathbb{R}, t > 0$.

Table 1. Unlabeled attachment accuracy scores for dependency parsing, using a bi-LSTM model (Kiperwasser & Goldberg, 2016). SparseMAP and its margin version, m-SparseMAP, produce the best parser on 4/5 datasets. For context, we include the scores of the CoNLL 2017 UDPipe baseline, which is trained under the same conditions (Straka & Straková, 2017).

Loss	en	zh	vi	ro	ja
Structured SVM	87.02	81.94	69.42	87.58	**96.24**
CRF	86.74	83.18	69.10	87.13	96.09
SparseMAP	86.90	**84.03**	69.71	87.35	96.04
m-SparseMAP	**87.34**	82.63	**70.87**	**87.63**	96.03
UDPipe baseline	87.68	82.14	69.63	87.36	95.94

Proof is given in Appendix C. Property 1 suggests that pmin-imizing $\ell_{\Omega, \boldsymbol{A}}$ aligns models with the true label. Property 2 shows how to compute subgradients of $\ell_{\Omega, \boldsymbol{A}}$ provided access to the inference output $[\boldsymbol{u}^\star; \boldsymbol{v}^\star] = \boldsymbol{A}\boldsymbol{y}^\star \in \mathbb{R}^k$. Combined with our efficient procedure described in Section 3.2, it makes the SparseMAP losses promising for structured prediction. Property 3 suggests that the strength of the penalty Ω can be adjusted by simply scaling $\boldsymbol{\eta}$. Finally, we remark that for a strongly-convex Ω, $\ell_{\Omega, \boldsymbol{A}}$ can be seen as a smoothed perceptron loss; other smoothed losses have been explored by Shalev-Shwartz & Zhang (2016).

5. Experimental Results

In this section, we experimentally validate SparseMAP on two natural language processing applications, illustrating the two main use cases presented: structured output prediction with the SparseMAP loss (§5.1) and structured hidden layers (§5.2). All models are implemented using the dynet library v2.0.2 (Neubig et al., 2017).

5.1. Dependency Parsing with the SparseMAP Loss

We evaluate the SparseMAP losses against the commonly used CRF and structured SVM losses. The task we focus on is non-projective *dependency parsing*: a structured output task consisting of predicting the directed tree of grammatical dependencies between words in a sentence (Jurafsky & Martin, 2018, Ch. 14). We use annotated Universal Dependency data (Nivre et al., 2016), as used in the CoNLL 2017 shared task (Zeman et al., 2017). To isolate the effect of the loss, we use the provided gold tokenization and part-of-speech tags. We follow closely the bidirectional LSTM arc-factored parser of Kiperwasser & Goldberg (2016), using the same model configuration; the only exception is not using externally pretrained embeddings. Parameters are trained using Adam (Kingma & Ba, 2015), tuning the learning rate on the grid $\{.5, 1, 2, 4, 8\} \times 10^{-3}$, expanded by a factor of 2 if the best model is at either end.

We experiment with 5 languages, diverse both in terms of

Table 2. Test accuracy scores for natural language inference with structured and unstructured variants of ESIM. In parentheses: the percentage of pairs of words with nonzero alignment scores.

ESIM variant	MultiNLI	SNLI
softmax	76.05 (100%)	86.52 (100%)
sequential	75.54 (13%)	**86.62** (19%)
matching	**76.13** (8%)	86.05 (15%)

family and in terms of the amount of training data (ranging from 1,400 sentences for Vietnamese to 12,525 for English). Test set results (Table 1) indicate that the SparseMAP losses outperform the SVM and CRF losses on 4 out of the 5 languages considered. This suggests that SparseMAP is a good middle ground between MAP-based and marginal-based losses in terms of smoothness and gradient sparsity.

Moreover, as illustrated in Figure 4, the SparseMAP loss encourages **sparse predictions**: models converge towards sparser solutions as they train, yielding very few ambiguous arcs. When confident, SparseMAP can predict a single tree. Otherwise, the small set of candidate parses returned can be easily visualized, often indicating genuine linguistic ambiguities (Figure 3). Returning a small set of parses, also sought concomitantly by Keith et al. (2018), is valuable in pipeline systems, *e.g.*, when the parse is an input to a downstream application: error propagation is diminished in cases where the highest-scoring tree is incorrect (which is the case for the sentences in Figure 3). Unlike K-best heuristics, SparseMAP dynamically adjusts its output sparsity, which is desirable on realistic data where most instances are easy.

5.2. Latent Structured Alignment for Natural Language Inference

In this section, we demonstrate SparseMAP for inferring latent structure in large-scale deep neural networks. We focus on the task of *natural language inference*, defined as the classification problem of deciding, given two sentences (a *premise* and a *hypothesis*), whether the premise *entails* the hypothesis, *contradicts* it, or is *neutral* with respect to it.

We consider novel structured variants of the state-of-the-art ESIM model (Chen et al., 2017). Given a premise P of length m and a hypothesis H of length n, ESIM:

1. Encodes P and H with an LSTM.

2. Computes alignment scores $\boldsymbol{G} \in \mathbb{R}^{m \times n}$; with g_{ij} the inner product between the P word i and H word j.

3. Computes P-to-H and H-to-P alignments using row-wise, respectively column-wise softmax on $\boldsymbol{G}$.

4. Augments P words with the weighted average of its aligned H words, and vice-versa.

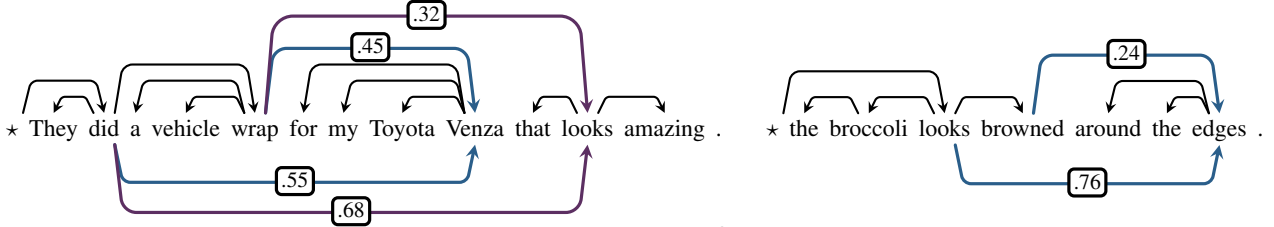5. Passes the result through another LSTM, then predicts.

Figure 3. Example of ambiguous parses from the UD English validation set. SparseMAP selects a small number of candidate parses (left: three, right: two), differing from each other in a small number of ambiguous dependency arcs. In both cases, the desired gold parse is among the selected trees (depicted by the arcs above the sentence), but it is not the highest-scoring one.
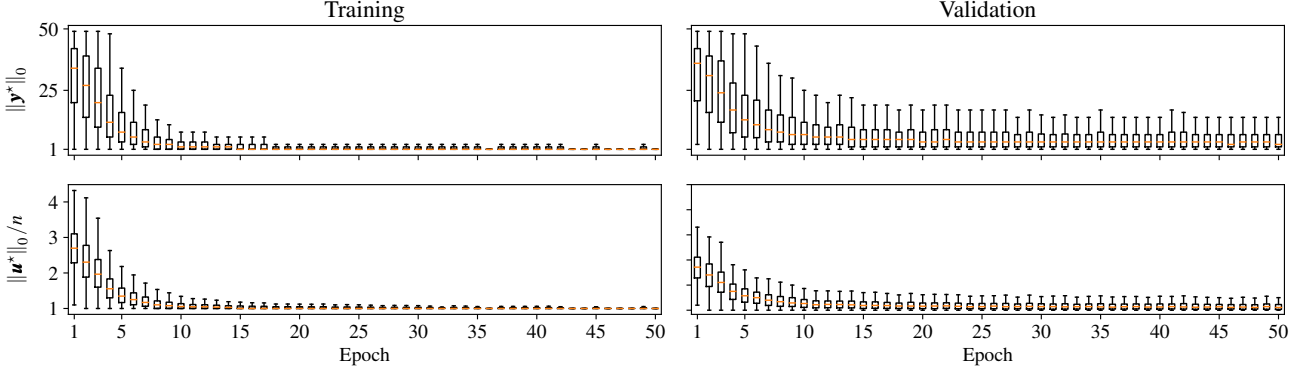

Figure 4. Distribution of the tree sparsity (top) and arc sparsity (bottom) of SparseMAP solutions during training on the Chinese dataset. Shown are respectively the number of trees and the average number of parents per word with nonzero probability.

We consider the following structured replacements for the independent row-wise and column-wise softmaxes (step 3):

Sequential alignment. We model the alignment of p to h as a sequence tagging instance of length m, with n possible tags corresponding to the n words of the hypothesis. Through *transition scores*, we enable the model to capture continuity and monotonicity of alignments: we parametrize transitioning from word t_1 to t_2 by binning the distance $t_2 - t_1$ into 5 groups, $\{-2 \text{ or less}, -1, 0, 1, 2 \text{ or more}\}$. We similarly parametrize the initial alignment using bins $\{1, 2 \text{ or more}\}$ and the final alignment as $\{-2 \text{ or less}, -1\}$, allowing the model to express whether an alignment starts at the beginning or ends on the final word of h; formally

$$\eta_F(i, t_1, t_2) := \begin{cases} w_{\text{bin}(t_2 - t_1)} & 0 < i < n, \\ w^{\text{start}}_{\text{bin}(t_2)} & i = 0, \\ w^{\text{end}}_{\text{bin}(t_1)} & i = n. \end{cases}$$

We align p to h applying the same method in the other direction, with different transition scores w. Overall, sequential alignment requires learning 18 additional scalar parameters.

Matching alignment. We now seek a symmetrical alignment in both directions simultaneously. To this end, we cast the alignment problem as finding a maximal weight bipartite matching. We recall from §2.2 that a solution can be found via the Hungarian algorithm (in contrast to marginal inference, which is #P-complete). When $n = m$, maximal

matchings can be represented as permutation matrices, and when $n \neq m$ some words remain unaligned. SparseMAP returns a weighted average of a few maximal matchings. This method requires no additional learned parameters.

We evaluate the two models alongside the softmax baseline on the SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2018) datasets.[3] All models are trained by SGD, with $0.9\times$ learning rate decay at epochs when the validation accuracy is not the best seen. We tune the learning rate on the grid $\{2^k : k \in \{-6, -5, -4, -3\}\}$, extending the range if the best model is at either end. The results in Table 2 show that structured alignments are competitive with softmax in terms of accuracy, but are orders of magnitude sparser. This sparsity allows them to produce global alignment structures that are interpretable, as illustrated in Figure 5.

Interestingly, we observe computational advantages of sparsity. Despite the overhead of GPU memory copying, both training and validation in our latent structure models take roughly the same time as with softmax and become faster as the models grow more certain. For the sake of comparison, Kim et al. (2017) report a $5\times$ slow-down in their structured attention networks, where they use marginal inference.

---

[3] We split the MultiNLI matched validation set into equal validation and test sets; for SNLI we use the provided split.
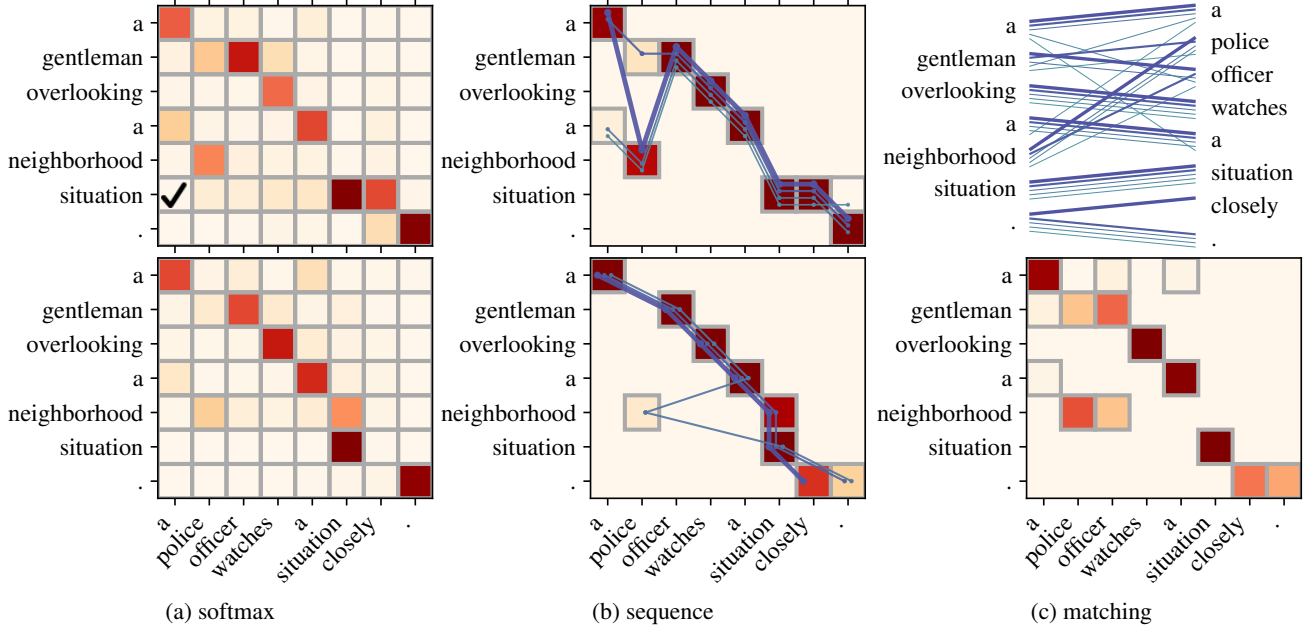
Figure 5. Latent alignments on an example from the SNLI validation set, correctly predicted as *neutral* by all compared models. The premise is on the y-axis, the hypothesis on the x-axis. Top: columns sum to 1; bottom: rows sum to 1. The matching alignment mechanism yields a symmetrical alignment, and is thus shown only once. Softmax yields a dense alignment (nonzero weights are marked with a border). The structures selected by sequential alignment are overlaid as paths; the selected matchings are displayed in the top right.

6. Related Work

Structured attention networks. Kim et al. (2017) and Liu & Lapata (2018) take advantage of the tractability of marginal inference in certain structured models and derive specialized backward passes for structured attention. In contrast, our approach is modular and general: with SparseMAP, the forward pass only requires MAP inference, and the backward pass is efficiently computed based on the forward pass results. Moreover, unlike marginal inference, SparseMAP yields sparse solutions, which is an appealing property statistically, computationally, and visually.

K-best inference. As it returns a small set of structures, SparseMAP brings to mind K-best inference, often used in pipeline NLP systems for increasing recall and handling uncertainty (Yang & Cardie, 2013). K-best inference can be approximated (or, in some cases, solved), roughly K times slower than MAP inference (Yanover & Weiss, 2004; Camerini et al., 1980; Chegireddy & Hamacher, 1987; Fromer & Globerson, 2009). The main advantages of SparseMAP are convexity, differentiablity, and modularity, as SparseMAP can be computed *in terms of* MAP subproblems. Moreover, it yields a distribution, unlike K-best, which does not reveal the gap between selected structures,

Learning permutations. A popular approach for differentiable permutation learning involves mean-entropic optimal transport relaxations (Adams & Zemel, 2011; Mena et al., 2018). Unlike SparseMAP, this does not apply to general

structures, and solutions are not directly expressible as combinations of a few permutations.

Regularized inference. Ravikumar et al. (2010), Meshi et al. (2015), and Martins et al. (2015) proposed ℓ_2 perturbations and penalties in various related ways, with the goal of solving LP-MAP approximate inference in graphical models. In contrast, the goal of our work is sparse structured prediction, which is not considered in the aforementioned work. Nevertheless, some of the formulations in their work share properties with SparseMAP; exploring the connections further is an interesting avenue for future work.

7. Conclusion

We introduced a new framework for sparse structured inference, SparseMAP, along with a corresponding loss function. We proposed efficient ways to compute the forward and backward passes of SparseMAP. Experimental results illustrate two use cases where sparse inference is well-suited. For structured prediction, the SparseMAP loss leads to strong models that make sparse, interpretable predictions, a good fit for tasks where local ambiguities are common, like many natural language processing tasks. For structured hidden layers, we demonstrated that SparseMAP leads to strong, interpretable networks trained end-to-end. Modular by design, SparseMAP can be applied readily to any structured problem for which MAP inference is available, including combinatorial problems such as linear assignment.

Acknowledgements

References

Adams, R. P. and Zemel, R. S. Ranking via sinkhorn propagation. *arXiv e-prints*, 2011.

Amos, B. and Kolter, J. Z. OptNet: Differentiable optimization as a layer in neural networks. In *ICML*, 2017.

Bakır, G., Hofmann, T., Schölkopf, B., Smola, A. J., Taskar, B., and Vishwanathan, S. V. N. *Predicting Structured Data*. The MIT Press, 2007.

Belanger, D. and McCallum, A. Structured prediction energy networks. In *ICML*, 2016.

Belanger, D., Sheldon, D., and McCallum, A. Marginal inference in MRFs using Frank-Wolfe. In *NIPS Workshop on Greedy Opt., FW and Friends*, 2013.

Belanger, D., Yang, B., and McCallum, A. End-to-end learning for structured prediction energy networks. In *ICML*, 2017.

Birkhoff, G. Tres observaciones sobre el algebra lineal. *Univ. Nac. Tucumán Rev. Ser. A*, 5:147–151, 1946.

Blondel, M., Martins, A. F., and Niculae, V. Learning classifiers with Fenchel-Young losses: Generalized entropies, margins, and algorithms. *arXiv e-prints*, 2018.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. A large annotated corpus for learning natural language inference. In *EMNLP*, 2015.

Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004.

Camerini, P. M., Fratta, L., and Maffioli, F. The k best spanning arborescences of a network. *Networks*, 10(2): 91–109, 1980.

Chegireddy, C. R. and Hamacher, H. W. Algorithms for finding K-best perfect matchings. *Discrete Applied Mathematics*, 18(2):155 – 165, 1987.

Chen, Q., Zhu, X., Ling, Z.-H., Wei, S., Jiang, H., and Inkpen, D. Enhanced LSTM for natural language inference. In *ACL*, 2017.

Chu, Y.-J. and Liu, T.-H. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400, 1965.

Clarke, F. H. *Optimization and Nonsmooth Analysis*. SIAM, 1990.

Collins, M. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *EMNLP*, 2002.

Domke, J. Learning graphical model parameters with approximate marginal inference. *IEEE T. Pattern. Anal.*, 35 (10):2454–2467, 2013.

Edmonds, J. Optimum branchings. *J. Res. Nat. Bur. Stand.*, 71B:233–240, 1967.

Fenchel, W. On conjugate convex functions. *Canad. J. Math*, 1(73-77), 1949.

Frank, M. and Wolfe, P. An algorithm for quadratic programming. *Nav. Res. Log.*, 3(1-2):95–110, 1956.

Fromer, M. and Globerson, A. An LP view of the M-best MAP problem. In *NIPS*, 2009.

Garber, D. and Meshi, O. Linear-memory and decomposition-invariant linearly convergent conditional gradient algorithm for structured polytopes. In *NIPS*, 2016.

Gimpel, K. and Smith, N. A. Softmax-margin CRFs: Training log-linear models with cost functions. In *NAACL*, 2010.

Jonker, R. and Volgenant, A. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, 1987.

Jurafsky, D. and Martin, J. H. *Speech and Language Processing (3rd ed.)*. draft, 2018.

Keith, K., Blodgett, S. L., and OâĂŹConnor, B. Monte Carlo syntax marginals for exploring and using dependency parses. In *NAACL*, 2018.

Kim, Y., Denton, C., Hoang, L., and Rush, A. M. Structured attention networks. In *ICLR*, 2017.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Kiperwasser, E. and Goldberg, Y. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*, 4:313–327, 2016.

Kirchhoff, G. Ueber die auflösung der gleichungen, auf welche man bei der untersuchung der linearen vertheilung galvanischer ströme geführt wird. *Annalen der Physik*, 148(12):497–508, 1847.

Koo, T., Globerson, A., Carreras Pérez, X., and Collins, M. Structured prediction models via the matrix-tree theorem. In *EMNLP*, 2007.

Krishnan, R. G., Lacoste-Julien, S., and Sontag, D. Barrier Frank-Wolfe for marginal inference. In *NIPS*, 2015.

Kschischang, F. R., Frey, B. J., and Loeliger, H.-A. Factor graphs and the sum-product algorithm. *IEEE T. Inform. Theory*, 47(2):498–519, 2001.

Kuhn, H. W. The Hungarian method for the assignment problem. *Nav. Res. Log.*, 2(1-2):83–97, 1955.

Lacoste-Julien, S. and Jaggi, M. On the global linear convergence of Frank-Wolfe optimization variants. In *NIPS*, 2015.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

Li, Z. and Eisner, J. First-and second-order expectation semirings with applications to minimum-risk training on translation forests. In *EMNLP*, 2009.

Liu, Y. and Lapata, M. Learning structured text representations. *TACL*, 6:63–75, 2018.

Martins, A. F. and Astudillo, R. F. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *ICML*, 2016.

Martins, A. F., Smith, N. A., and Xing, E. P. Concise integer linear programming formulations for dependency parsing. In *ACL-IJCNLP*, 2009.

Martins, A. F., Figueiredo, M. A., Aguiar, P. M., Smith, N. A., and Xing, E. P. AD3: Alternating directions dual decomposition for MAP inference in graphical models. *JMLR*, 16(1):495–545, 2015.

McDonald, R., Crammer, K., and Pereira, F. Online large-margin training of dependency parsers. In *ACL*, 2005.

Mena, G., Belanger, D., Linderman, S., and Snoek, J. Learning latent permutations with Gumbel-Sinkhorn networks. In *ICLR*, 2018.

Mensch, A. and Blondel, M. Differentiable dynamic programming for structured prediction and attention. In *ICML*, 2018.

Meshi, O., Mahdavi, M., and Schwing, A. Smooth and strong: MAP inference with linear convergence. In *NIPS*, 2015.

Neubig, G., Dyer, C., Goldberg, Y., Matthews, A., Ammar, W., Anastasopoulos, A., Ballesteros, M., Chiang, D., Clothiaux, D., Cohn, T., et al. DyNet: The dynamic neural network toolkit. *preprint arXiv:1701.03980*, 2017.

Niculae, V. and Blondel, M. A regularized framework for sparse and structured neural attention. In *NIPS*, 2017.

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., McDonald, R. T., Petrov, S., Pyysalo, S., Silveira, N., et al. Universal Dependencies v1: A multilingual treebank collection. In *LREC*, 2016.

Nocedal, J. and Wright, S. *Numerical Optimization*. Springer New York, 1999.

Nowozin, S., Gehler, P. V., Jancsary, J., and Lampert, C. H. *Advanced Structured Prediction*. MIT Press, 2014.

Rabiner, L. R. A tutorial on Hidden Markov Models and selected applications in speech recognition. *P. IEEE*, 77 (2):257–286, 1989.

Ravikumar, P., Agarwal, A., and Wainwright, M. J. Message-passing for graph-structured linear programs: Proximal methods and rounding schemes. *JMLR*, 11:1043–1080, 2010.

Shalev-Shwartz, S. and Zhang, T. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Math. Program.*, 155(1):105–145, 2016.

Smith, D. A. and Smith, N. A. Probabilistic models of nonprojective dependency trees. In *EMNLP*, 2007.

Smith, N. A. *Linguistic Structure Prediction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool, May 2011.

Stoyanov, V., Ropson, A., and Eisner, J. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *AISTATS*, 2011.

Straka, M. and Straková, J. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *CoNLL Shared Task*, 2017.

Taskar, B. *Learning Structured Prediction Models: A Large Margin Approach*. PhD thesis, Stanford University, 2004.

Taskar, B., Guestrin, C., and Koller, D. Max-Margin Markov Networks. In *NIPS*, 2003.

Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

Valiant, L. G. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8(2):189–201, 1979.

Vinyes, M. and Obozinski, G. Fast column generation for atomic norm regularization. In *AISTATS*, 2017.

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

Williams, A., Nangia, N., and Bowman, S. R. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*, 2018.

Wolfe, P. Finding the nearest point in a polytope. *Mathematical Programming*, 11(1):128–149, 1976.

Yang, B. and Cardie, C. Joint inference for fine-grained opinion extraction. In *ACL*, 2013.

Yanover, C. and Weiss, Y. Finding the M most probable configurations using loopy belief propagation. In *NIPS*, 2004.

Zeman, D., Popel, M., Straka, M., Hajic, J., Nivre, J., Ginter, F., Luotolahti, J., Pyysalo, S., Petrov, S., Potthast, M., et al. CoNLL 2017 shared task: Multilingual parsing from raw text to universal dependencies. *CoNLL*, 2017.

Supplementary material

A. Implementation Details for SparseMAP Solvers

A.1. Conditional Gradient Variants

We adapt the presentation of vanilla, away-step and pairwise conditional gradient of Lacoste-Julien & Jaggi (2015).

Recall the SparseMAP optimization problem (Equation 5), which we rewrite below as a minimization, to align with the formulation in (Lacoste-Julien & Jaggi, 2015)

$$\text{SparseMAP}_{\boldsymbol{A}}(\boldsymbol{\eta}) \coloneqq \underset{\boldsymbol{u}:\ [\boldsymbol{u},\boldsymbol{v}]\in\mathcal{M}_{\boldsymbol{A}}}{\arg\min}\ f(\boldsymbol{u},\boldsymbol{v}), \qquad \text{where } f(\boldsymbol{u},\boldsymbol{v}) \coloneqq \frac{1}{2}\|\boldsymbol{u}\|_2^2 - \boldsymbol{\eta}_U^\top\boldsymbol{u} - \boldsymbol{\eta}_F^\top\boldsymbol{v}.$$

The gradients of the objective function f w.r.t. the two variables are

$$\nabla_{\boldsymbol{u}}f(\boldsymbol{u}',\boldsymbol{v}') = \boldsymbol{u}' - \boldsymbol{\eta}_U, \qquad \nabla_{\boldsymbol{v}}f(\boldsymbol{u}',\boldsymbol{v}') = -\boldsymbol{\eta}_V.$$

The ingredients required to apply conditional gradient algorithms are solving linear minimization problem, selecting the away step, computing the Wolfe gap, and performing line search.

Linear minimization problem. For SparseMAP, this amounts to a MAP inference call, since

$$\underset{[\boldsymbol{u},\boldsymbol{v}]\in\mathcal{M}_{\boldsymbol{A}}}{\arg\min}\ \langle\nabla_{\boldsymbol{u}}f(\boldsymbol{u}',\boldsymbol{v}'),\boldsymbol{u}\rangle + \langle\nabla_{\boldsymbol{v}}f(\boldsymbol{u}',\boldsymbol{v}'),\boldsymbol{v}\rangle$$
$$= \underset{[\boldsymbol{u},\boldsymbol{v}]\in\mathcal{M}_{\boldsymbol{A}}}{\arg\min}\ (\boldsymbol{u}'-\boldsymbol{\eta}_U)^\top\boldsymbol{u} - \boldsymbol{\eta}_F^\top\boldsymbol{v}$$
$$= \{[\boldsymbol{m}_s;\boldsymbol{n}_s]\ :\ s\in\text{MAP}_{\boldsymbol{A}}(\boldsymbol{\eta}_U - \boldsymbol{u}',\boldsymbol{\eta}_F)\}.$$

where we assume $\text{MAP}_{\boldsymbol{A}}$ yields the set of maximally-scoring structures.

Away step selection. This step involves searching the currently selected structures in the active set $\mathcal{I}$ with the *opposite* goal: finding the structure *maximizing* the linearization

$$\underset{s\in\mathcal{I}}{\arg\max}\ \langle\nabla_{\boldsymbol{u}}f(\boldsymbol{u}',\boldsymbol{v}'),\boldsymbol{m}_s\rangle + \langle\nabla_{\boldsymbol{v}}f(\boldsymbol{u}',\boldsymbol{v}'),\boldsymbol{n}_s\rangle$$
$$= \underset{s\in\mathcal{I}}{\arg\max}\ (\boldsymbol{u}'-\boldsymbol{\eta}_U)^\top\boldsymbol{m}_s - \boldsymbol{\eta}_F^\top\boldsymbol{n}_s$$

Wolfe gap. The gap at a point $\boldsymbol{d} = [\boldsymbol{d}_{\boldsymbol{u}};\boldsymbol{d}_{\boldsymbol{v}}]$ is given by

$$\begin{aligned}
\text{gap}(\boldsymbol{d},\boldsymbol{u}') &\coloneqq \langle-\nabla_{\boldsymbol{u}}f(\boldsymbol{u}',\boldsymbol{v}'),\boldsymbol{d}_{\boldsymbol{u}}\rangle + \langle-\nabla_{\boldsymbol{v}}f(\boldsymbol{u}',\boldsymbol{v}'),\boldsymbol{d}_{\boldsymbol{v}}\rangle \\
&= \langle\boldsymbol{\eta}_U - \boldsymbol{u}',\boldsymbol{d}_{\boldsymbol{u}}\rangle + \langle\boldsymbol{\eta}_F,\boldsymbol{d}_{\boldsymbol{v}}\rangle.
\end{aligned} \tag{7}$$

Line search. Once we have picked a direction $\boldsymbol{d} = [\boldsymbol{d}_{\boldsymbol{u}};\boldsymbol{d}_{\boldsymbol{v}}]$, we can pick the optimal step size by solving a simple optimization problem. Let $\boldsymbol{u}_\gamma \coloneqq \boldsymbol{u}' + \gamma\boldsymbol{d}_{\boldsymbol{u}}$, and $\boldsymbol{v}_\gamma \coloneqq \boldsymbol{v}' + \gamma\boldsymbol{d}_{\boldsymbol{v}}$. We seek γ so as to optimize

$$\underset{\gamma\in[0,\gamma_{\max}]}{\arg\min}\ f(\boldsymbol{u}_\gamma,\boldsymbol{v}_\gamma)$$

Setting the gradient w.r.t. γ to 0 yields

$$\begin{aligned}
0 &= \frac{\partial}{\partial\gamma}f(\boldsymbol{u}_\gamma,\boldsymbol{v}_\gamma) \\
&= \langle\boldsymbol{d}_{\boldsymbol{u}},\nabla_{\boldsymbol{u}}f(\boldsymbol{u}_\gamma,\boldsymbol{v}_\gamma)\rangle + \langle\boldsymbol{d}_{\boldsymbol{v}},\nabla_{\boldsymbol{v}}f(\boldsymbol{u}_\gamma,\boldsymbol{v}_\gamma)\rangle \\
&= \langle\boldsymbol{d}_{\boldsymbol{u}},\boldsymbol{u}' + \gamma\boldsymbol{d}_{\boldsymbol{u}} - \boldsymbol{\eta}_U\rangle + \langle\boldsymbol{d}_{\boldsymbol{v}},-\boldsymbol{\eta}_F\rangle \\
&= \gamma\|\boldsymbol{d}_{\boldsymbol{u}}\|_2^2 + \boldsymbol{u}'^\top\boldsymbol{d}_{\boldsymbol{u}} - \boldsymbol{\eta}^\top\boldsymbol{d}
\end{aligned}$$

We may therefore compute the optimal step size γ as

$$\gamma = \max\left(0, \min\left(\gamma_{\text{max}}, \frac{\boldsymbol{\eta}^\top \boldsymbol{d} - \boldsymbol{u}'^\top \boldsymbol{d_u}}{\|\boldsymbol{d_u}\|_2^2}\right)\right) \tag{8}$$

Algorithm 1 Conditional gradient for SparseMAP

1: Initialization: $\quad s^{(0)} \leftarrow \text{MAP}_{\boldsymbol{A}}(\boldsymbol{\eta}_U, \boldsymbol{\eta}_F); \quad \mathcal{I}^{(0)} = \{s^{(0)}\}; \quad \boldsymbol{y}^{(0)} = \boldsymbol{e}_{s^{(0)}}; \quad [\boldsymbol{u}^{(0)}; \boldsymbol{v}^{(0)}] = \boldsymbol{a}_{s^{(0)}}$
2: **for** $t = 0 \ldots t_{\text{max}}$ **do**
3: $\quad s \leftarrow \text{MAP}_{\boldsymbol{A}}(\boldsymbol{\eta}_U - \boldsymbol{u}^{(t)}, \boldsymbol{\eta}_F); \qquad\qquad \boldsymbol{d}^{\text{F}} \leftarrow \boldsymbol{a}_s - [\boldsymbol{u}^{(t)}; \boldsymbol{v}^{(t)}] \quad$ (forward direction)
4: $\quad w \leftarrow \underset{w \in \mathcal{I}^{(t)}}{\arg\max} \; (\boldsymbol{\eta}_U - \boldsymbol{u}^{(t)})^\top \boldsymbol{m}_w + \boldsymbol{\eta}_F^\top \boldsymbol{n}_w; \qquad \boldsymbol{d}^{\text{W}} \leftarrow [\boldsymbol{u}^{(t)}; \boldsymbol{v}^{(t)}] - \boldsymbol{a}_w \quad$ (away direction)
5: $\quad$ **if** $\text{gap}(\boldsymbol{d}^{\text{F}}, \boldsymbol{u}^{(t)}) < \epsilon$ **then**
6: $\qquad$ **return** $\boldsymbol{u}^{(t)} \quad$ (Equation 7)
7: $\quad$ **end if**
8: $\quad$ **if** variant $=$ vanilla **then**
9: $\qquad \boldsymbol{d} \leftarrow \boldsymbol{d}^{\text{F}}; \qquad \gamma_{\text{max}} \leftarrow 1$
10: $\quad$ **else if** variant $=$ pairwise **then**
11: $\qquad \boldsymbol{d} \leftarrow \boldsymbol{d}^{\text{F}} + \boldsymbol{d}^{\text{W}}; \qquad \gamma_{\text{max}} \leftarrow y_w$
12: $\quad$ **else if** variant $=$ away-step **then**
13: $\qquad$ **if** $\text{gap}(\boldsymbol{d}^{\text{F}}, \boldsymbol{u}^{(t)}) \geq \text{gap}(\boldsymbol{d}^{\text{W}}, \boldsymbol{u}^{(t)})$ **then**
14: $\qquad\quad \boldsymbol{d} \leftarrow \boldsymbol{d}^{\text{F}}; \qquad \gamma_{\text{max}} \leftarrow 1$
15: $\qquad$ **else**
16: $\qquad\quad \boldsymbol{d} \leftarrow \boldsymbol{d}^{\text{A}}; \qquad \gamma_{\text{max}} \leftarrow y_w/(1 - y_w)$
17: $\qquad$ **end if**
18: $\quad$ **end if**
19: $\quad$ Compute step size $\gamma \quad$ (Equation 8)
20: $\quad [\boldsymbol{u}^{(t+1)}; \boldsymbol{v}^{(t+1)}] \leftarrow [\boldsymbol{u}^{(t)}; \boldsymbol{v}^{(t)}] + \boldsymbol{d}$
21: $\quad$ Update $\mathcal{I}^{(t+1)}$ and $\boldsymbol{y}^{(t+1)}$ accordingly.
22: **end for**

A.2. The Active Set Algorithm

We use a variant of the active set algorithm (Nocedal & Wright, 1999, Ch. 16.4 & 16.5) as proposed for the quadratic subproblems of the AD^3 algorithm; our presentation follows (Martins et al., 2015, Algorithm 3). At each step, the active set algorithm solves a relaxed variant of the SparseMAP QP, relaxing the non-negativity constraint on $\boldsymbol{y}$, and restricting the solution to the current active set $\mathcal{I}$

$$\text{minimize}_{\boldsymbol{y}_\mathcal{I} \in \mathbb{R}^{|\mathcal{I}|}} \quad \frac{1}{2} \|\boldsymbol{M}_\mathcal{I} \boldsymbol{y}_\mathcal{I}\|_2^2 - \boldsymbol{\eta}^\top \boldsymbol{A}_\mathcal{I} \boldsymbol{y}_\mathcal{I} \qquad\qquad \text{subject to} \quad \boldsymbol{1}^\top \boldsymbol{y}_\mathcal{I} = 1$$

whose solution can be found by solving the KKT system

$$\begin{bmatrix} \boldsymbol{M}_\mathcal{I}^\top \boldsymbol{M}_\mathcal{I} & \boldsymbol{1} \\ \boldsymbol{1}^\top & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_\mathcal{I} \\ \tau \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_\mathcal{I}^\top \boldsymbol{\eta} \\ 1 \end{bmatrix}. \tag{9}$$

At each iteration, the (symmetric) design matrix in Equation 9 is updated by adding or removing a row and a column; therefore its inverse (or a decomposition) may be efficiently maintained and updated.

Line search. The optimal step size for moving a feasible current estimate $\boldsymbol{y}'$ toward a solution $\hat{\boldsymbol{y}}$ of Equation 9, while keeping feasibility, is given by (Martins et al., 2015, Equation 31)

$$\gamma = \min\left(1, \min_{s \in \mathcal{I}, \, y'_s > \hat{y}_s} \frac{y'_s}{y'_s - \hat{y}_s}\right) \tag{10}$$

When $\gamma \leq 1$ this update zeros out a coordinate of $\boldsymbol{y}'$; otherwise, $\mathcal{I}$ remains the same.

Algorithm 2 Active Set algorithm for SparseMAP

1: Initialization: $s^{(0)} \leftarrow \mathsf{MAP}_{\boldsymbol{A}}(\boldsymbol{\eta}_U, \boldsymbol{\eta}_F)$; $\quad \mathcal{I}^{(0)} = \{s^{(0)}\}$; $\quad \boldsymbol{y}^{(0)} = \boldsymbol{e}_{s^{(0)}}$; $\quad [\boldsymbol{u}^{(0)}; \boldsymbol{v}^{(0)}] = \boldsymbol{a}_{s^{(0)}}$
2: **for** $t = 0 \ldots t_{\max}$ **do**
3: $\quad$ Solve the relaxed QP restricted to $\mathcal{I}^{(t)}$; get $\hat{\boldsymbol{y}}, \hat{\tau}, \hat{\boldsymbol{u}} = \boldsymbol{M}\hat{\boldsymbol{y}}$ $\quad$ (Equation 9)
4: $\quad$ **if** $\hat{\boldsymbol{y}} = \boldsymbol{y}^{(t)}$ **then**
5: $\quad\quad$ $s \leftarrow \mathsf{MAP}_{\boldsymbol{A}}(\boldsymbol{\eta}_U - \hat{\boldsymbol{u}}, \boldsymbol{\eta}_F)$
6: $\quad\quad$ **if** $\mathrm{gap}(\boldsymbol{a}_s, \hat{\boldsymbol{u}}) \leq \hat{\tau}$ **then**
7: $\quad\quad\quad$ **return** $\boldsymbol{u}^{(t)}$ $\quad$ (Equation 7)
8: $\quad\quad$ **else**
9: $\quad\quad\quad$ $\mathcal{I}^{(t+1)} \leftarrow \mathcal{I}^{(t)} \cup \{s\}$
10: $\quad\quad$ **end if**
11: $\quad$ **else**
12: $\quad\quad$ Compute step size γ $\quad$ (Equation 10)
13: $\quad\quad$ $\boldsymbol{y}^{(t+1)} \leftarrow (1 - \gamma)\boldsymbol{y}^{(t)} + \gamma\hat{\boldsymbol{y}}$ $\quad$ (sparse update)
14: $\quad\quad$ Update $\mathcal{S}^{(t+1)}$ if necessary
15: $\quad$ **end if**
16: **end for**

B. Computing the SparseMAP Jacobian: Proof of Proposition 1

Recall that SparseMAP is defined as the $\boldsymbol{u}^\star$ that maximizes the value of the quadratic program (Equation 5),

$$g(\boldsymbol{\eta}_U, \boldsymbol{\eta}_F) := \max_{[\boldsymbol{u}; \boldsymbol{v}] \in \mathcal{M}_{\boldsymbol{A}}} \boldsymbol{\eta}_U^\top \boldsymbol{u} + \boldsymbol{\eta}_F^\top \boldsymbol{v} - \frac{1}{2} \|\boldsymbol{u}\|_2^2. \tag{11}$$

As the ℓ_2^2 norm is strongly convex, there is always a unique minimizer $\boldsymbol{u}^\star$ (implying that SparseMAP is well-defined), and the convex conjugate of the QP in (11), $g^\star(\boldsymbol{u}, \boldsymbol{v}) = \{\frac{1}{2} \|\boldsymbol{u}\|_2^2, [\boldsymbol{u}; \boldsymbol{v}] \in \mathcal{M}_{\boldsymbol{A}}; -\infty \text{ otherwise}\}$ is smooth in $\boldsymbol{u}$, implying that SparseMAP (which only returns $\boldsymbol{u}$) is Lipschitz-continuous and thus differentiable almost everywhere.

We now rewrite the QP in Equation 11 in terms of the convex combination of vertices of the marginal polytope

$$\min_{\boldsymbol{y} \in \triangle^D} \frac{1}{2} \|\boldsymbol{M}\boldsymbol{y}\|_2^2 - \boldsymbol{\theta}^\top \boldsymbol{y} \qquad \text{where } \boldsymbol{\theta} := \boldsymbol{A}^\top \boldsymbol{\eta} \tag{12}$$

We use the optimality conditions of problem 12 to derive an explicit relationship between $\boldsymbol{u}^\star$ and $\boldsymbol{x}$. At an optimum, the following KKT conditions hold

$$\boldsymbol{M}^\top \boldsymbol{M} \boldsymbol{y}^\star - \boldsymbol{\lambda}^\star + \tau^\star \mathbf{1} = \boldsymbol{\theta} \tag{13}$$

$$\mathbf{1}^\top \boldsymbol{y}^\star = 1 \tag{14}$$

$$\boldsymbol{y}^\star \geq \mathbf{0} \tag{15}$$

$$\boldsymbol{\lambda}^\star \geq \mathbf{0} \tag{16}$$

$$\boldsymbol{\lambda}^{\star\top} \boldsymbol{y}^\star = 0 \tag{17}$$

Let $\mathcal{I}$ denote the support of $\boldsymbol{y}^\star$, *i.e.*, $\mathcal{I} = \{s \; : \; y_s^\star > 0\}$. From Equation 17 we have $\boldsymbol{\lambda}_{\mathcal{I}} = \mathbf{0}$ and therefore

$$\boldsymbol{M}_{\mathcal{I}}^\top \boldsymbol{M}_{\mathcal{I}} \boldsymbol{y}_{\mathcal{I}}^\star + \tau^\star \mathbf{1} = \boldsymbol{\theta}_{\mathcal{I}} \tag{18}$$

$$\mathbf{1}^\top \boldsymbol{y}_{\mathcal{I}}^\star = 1 \tag{19}$$

Solving for $\boldsymbol{y}_{\mathcal{I}}^\star$ in Equation 18 we get a direct expression

$$\boldsymbol{y}_{\mathcal{I}}^\star = (\boldsymbol{M}_{\mathcal{I}}^\top \boldsymbol{M}_{\mathcal{I}})^{-1}(\boldsymbol{\theta}_{\mathcal{I}} - \tau^\star \mathbf{1}) = \boldsymbol{Z}(\boldsymbol{\theta}_{\mathcal{I}} - \tau^\star \mathbf{1}).$$

where we introduced $\boldsymbol{Z} = (\boldsymbol{M}^\top \boldsymbol{M})^{-1}$. Solving for $\tau^\star$ yields

$$\tau^\star = \frac{1}{\mathbf{1}^T \boldsymbol{Z} \mathbf{1}} \left(\mathbf{1}^T \boldsymbol{Z} \boldsymbol{\theta}_{\mathcal{I}} - 1 \right)$$

Plugging this back and left-multiplying by $\boldsymbol{M}_{\mathcal{I}}$ we get

$$\boldsymbol{u}^\star = \boldsymbol{M}_{\mathcal{I}} \boldsymbol{y}_{\mathcal{I}}^\star = \boldsymbol{M}_{\mathcal{I}} \boldsymbol{Z} \left(\boldsymbol{\theta}_{\mathcal{I}} - \frac{1}{\mathbf{1}^\top \boldsymbol{Z} \mathbf{1}} \mathbf{1}^\top \boldsymbol{Z} \boldsymbol{\theta}_{\mathcal{I}} \mathbf{1} + \frac{1}{\mathbf{1}^\top \boldsymbol{Z} \mathbf{1}} \mathbf{1} \right)$$

Note that, in a neighborhood of $\boldsymbol{\eta}$, the support of the solution $\mathcal{I}$ is constant. (On the measure-zero set of points where the support changes, SparseMAP is subdifferentiable and our assumption yields a generalized Jacobian (Clarke, 1990).) Differentiating w.r.t. the score of a configuration θ_s, we get the expression

$$\frac{\partial \boldsymbol{u}^\star}{\partial \theta_s} = \begin{cases} \boldsymbol{M} \left(\boldsymbol{I} - \frac{1}{\mathbf{1}^T \boldsymbol{Z} \mathbf{1}} \boldsymbol{Z} \mathbf{1} \mathbf{1}^T \right) \boldsymbol{z}_s & s \in \mathcal{I} \\ \boldsymbol{0} & s \notin \mathcal{I} \end{cases} \tag{20}$$

Since $\theta_s = \boldsymbol{a}_s^\top \boldsymbol{\eta}$, by the chain rule, we get the desired result

$$\frac{\partial \boldsymbol{u}^\star}{\partial \boldsymbol{\eta}} = \frac{\partial \boldsymbol{u}^\star}{\partial \boldsymbol{\theta}} \boldsymbol{A}^\top. \tag{21}$$

C. Fenchel-Young Losses: Proof of Proposition 2

We recall that the structured Fenchel-Young loss defined by a convex $\Omega : \mathbb{R}^D \to \mathbb{R}$ and a matrix $\boldsymbol{A}$ is defined as

$$\ell_{\Omega, \boldsymbol{A}} : \mathbb{R}^k \times \triangle^D \to \mathbb{R}, \quad \ell_{\Omega, \boldsymbol{A}}(\boldsymbol{\eta}, \boldsymbol{y}) := \Omega_\triangle^*(\boldsymbol{A}^\top \boldsymbol{\eta}) + \Omega_\triangle(\boldsymbol{y}) - \boldsymbol{\eta}^\top \boldsymbol{A} \boldsymbol{y}.$$

Since $\Omega_\triangle$ is the restriction of a convex function to a convex set, it is convex (Boyd & Vandenberghe, 2004, Section 3.1.2).

Property 1. From the Fenchel-Young inequality (Fenchel, 1949; Boyd & Vandenberghe, 2004, Section 3.3.2), we have

$$\boldsymbol{\theta}^\top \boldsymbol{y} \le \Omega_\triangle^*(\boldsymbol{\theta}) + \Omega_\triangle(\boldsymbol{y}).$$

In particular, when $\boldsymbol{\theta} = \boldsymbol{A}^\top \boldsymbol{\eta}$,

$$0 \le -\boldsymbol{\eta}^\top \boldsymbol{A} \boldsymbol{y} + \Omega_\triangle^*(\boldsymbol{A}^\top \boldsymbol{\eta}) + \Omega_\triangle(\boldsymbol{y})$$
$$= \ell_{\Omega, \boldsymbol{A}}(\boldsymbol{\eta}, \boldsymbol{y}).$$

Equality is achieved when

$$\Omega_\triangle^*(\boldsymbol{A}^\top \boldsymbol{\eta}) = \boldsymbol{\eta}^\top \boldsymbol{A} \boldsymbol{y} - \Omega_\triangle(\boldsymbol{y}) \iff$$
$$\max_{\boldsymbol{y}' \in \triangle^d} \boldsymbol{\eta}^\top \boldsymbol{A} \boldsymbol{y}' - \Omega(\boldsymbol{y}') = \boldsymbol{\eta}^\top \boldsymbol{A} \boldsymbol{y} - \Omega(\boldsymbol{y}),$$

where we used the fact that $\boldsymbol{y} \in \triangle^d$. The second part of the claim follows.

Property 2. To prove convexity in $\boldsymbol{\eta}$, we rewrite the loss, for fixed $\boldsymbol{y}$, as

$$\ell_{\Omega, \boldsymbol{A}}(\boldsymbol{\eta}) = h(\boldsymbol{A}^\top \boldsymbol{\eta}) + \text{const}, \quad \text{where} \quad h(\boldsymbol{\theta}) = \Omega_\triangle^*(\boldsymbol{\theta}) - \boldsymbol{\theta}^\top \boldsymbol{y}.$$

$\Omega_\triangle^*$ is a convex conjugate, and thus itself convex. Linear functions are convex, and the sum of two convex functions is convex, therefore h is convex. Finally, the composition of a convex function with a linear function is convex as well, thus the function $(h \boldsymbol{A}^\top)$ is convex. Convexity of $\ell_{\Omega, \boldsymbol{A}}$ in $\boldsymbol{\eta}$ directly follows. Convexity in $\boldsymbol{y}$ is straightforward, as the sum of a convex and a linear function (Boyd & Vandenberghe, 2004, Sections 3.2.1, 3.2.2, 3.3.1).

Property 3. This follows from the scaling property of the convex conjugate (Boyd & Vandenberghe, 2004, Section 3.3.2)

$$(t\Omega)^*(\boldsymbol{\theta}) = t\Omega^*(t^{-1} \boldsymbol{\theta})$$

Denoting $\boldsymbol{\eta}' = t^{-1} \boldsymbol{\eta}$, we have that

$$\ell_{t\Omega, \boldsymbol{A}}(\boldsymbol{\eta}, \boldsymbol{y}) = (t\Omega_\triangle)^*(\boldsymbol{A}^\top \boldsymbol{\eta}) + t\Omega_\triangle(\boldsymbol{y}) - \boldsymbol{\eta}^\top \boldsymbol{A} \boldsymbol{y}$$
$$= t\Omega_\triangle^*(\boldsymbol{A}^\top \boldsymbol{\eta}') + t\Omega_\triangle(\boldsymbol{y}) - \boldsymbol{\eta}^\top \boldsymbol{A} \boldsymbol{y}$$
$$= t\left(\Omega_\triangle^*(\boldsymbol{A}^\top \boldsymbol{\eta}') + \Omega_\triangle(\boldsymbol{y}) - \boldsymbol{\eta}'^\top \boldsymbol{A} \boldsymbol{y}\right) = t\ell_{\Omega, \boldsymbol{A}}(t^{-1} \boldsymbol{\eta}, \boldsymbol{y}).$$