Understanding the Mechanics of SPIGOT: Surrogate Gradients for Latent Structure Learning

Tsvetomila Mihaylova
Instituto de Telecomunicações
Instituto Superior Técnico
Lisbon, Portugal
tsvetomila.mihaylova@lx.it.pt

Vlad Niculae[†]
Informatics Institute
University of Amsterdam
The Netherlands
v.niculae@uva.nl

André F. T. Martins
Instituto de Telecomunicações
LUMLIS (Lisbon ELLIS Unit)
Instituto Superior Técnico & Unbabel
Lisbon, Portugal
andre.t.martins@tecnico.ulisboa.com

Abstract

Latent structure models are a powerful tool for modeling language data: they can mitigate the error propagation and annotation bottleneck in pipeline systems, while simultaneously uncovering linguistic insights about the data. One challenge with end-to-end training of these models is the argmax operation, which has null gradient. In this paper, we focus on *surrogate gradients*, a popular strategy to deal with this problem. We explore latent structure learning through the angle of *pulling back* the downstream learning objective. In this paradigm, we discover a principled motivation for both the straight-through estimator (STE) as well as the recently-proposed SPIGOT—a variant of STE for structured models. Our perspective leads to new algorithms in the same family. We empirically compare the known and the novel pulled-back estimators against the popular alternatives, yielding new insight for practitioners and revealing intriguing failure cases.

1 Introduction

Natural language data is *rich in structure*, but most of the structure is not visible at the surface. Machine learning models tackling high-level language tasks would benefit from uncovering underlying structures such as trees, sequence tags, or segmentations. Traditionally, practitioners turn to *pipeline* approaches where an external, pretrained model is used to predict, *e.g.*, syntactic structure. The benefit of this approach is that the predicted tree is readily available for inspection, but the downside is that the errors can easily propagate throughout the pipeline and require further attention (Finkel et al., 2006; Sutton and McCallum, 2005; Toutanova, 2005). In contrast, deep neural architectures tend to eschew such preprocessing, and instead learn soft hidden representations, not easily amenable to visualization and analysis.

The best of both worlds would be to *model structure as a latent variable*, combining the transparency of the pipeline approach with the end-to-end unsupervised representation learning that makes deep models appealing. Moreover, large-capacity model tend to rediscover structure from scratch (Tenney et al., 2019), so structured latent variables may reduce the required capacity.

Learning with discrete, combinatorial latent variables is, however, challenging, due to the intersection of *large cardinality* and *null gradient* issues. For example, when learning a latent dependency tree, the latent parser must choose among an exponentially large set of possible trees; what's more, the parser may only learn from gradient information from the downstream task. If the highest-scoring tree is selected using an *argmax* operation, the gradients will be zero, preventing learning.

One strategy for dealing with the null gradient issue is to use a **surrogate gradient**, explicitly overriding the zero gradient from the chain rule, as if a different computation had been performed. The most commonly known example is the *straight-through estimator* (STE; Bengio et al., 2013), which pretends that the *argmax* node was instead an *identity* operator. Such methods lead to a fundamental mismatch between the objective and the learning algorithm. The effect of this mismatch is still insufficiently understood, and the design of successful new variants is therefore challenging. For example, the recently-proposed SPIGOT method (Peng et al., 2018) found it beneficial to use a projection as part of the surrogate gradient.

In this paper, we study surrogate gradient methods for deterministic learning with discrete structured latent variables. Our contributions are:

- We propose a novel motivation for surrogate gra-

[†]Work partially done while VN was at the Instituto de Telecomunicações, Lisbon.

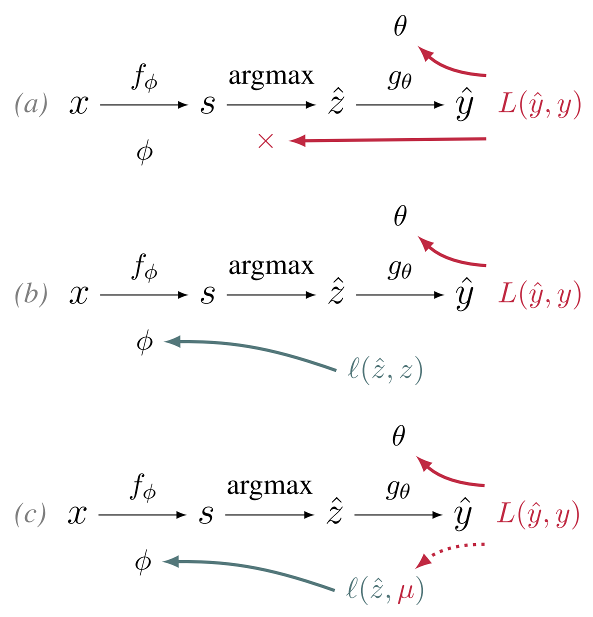

Figure 1: A model with a discrete latent variable z. Given an input x, we assign a score $s_z = [f(x)]_z$ to each choice, and pick the highest scoring one, $\hat{z}$, to predict $\hat{y} = g_\theta(\hat{z})$. For simplicity, here g_θ does not access x directly. (a). Since argmax has null gradients, the encoder parameters ϕ do not receive updates. (b). If ground truth supervision were available for the latent z, ϕ could be trained jointly with an auxiliary loss. (c). As such supervision is not available, we induce a best-guess label μ by pulling back the downstream loss. This strategy recovers the STE and SPIGOT estimators.

dient methods, based on optimizing a **pulled-back loss**, thereby inducing pseudo-supervision on the latent variable. This leads to new insight into both STE and SPIGOT.

- We show how our framework may be used to derive new surrogate gradient methods, by varying the loss function or the inner optimization algorithm used for inducing the pseudo-supervision.

- We experimentally validate our discoveries on a controllable experiment as well as on English-language sentiment analysis and natural language inference, comparing against stochastic and relaxed alternatives, yielding new insights, and identifying noteworthy failure cases.

While the discrete methods do not outperform the relaxed alternatives using the same building blocks, we hope that our interpretation and insights would trigger future latent structure research.

The code for the paper is available on `https://github.com/deep-spin/understanding-spigot`.

2 Related Work

Discrete latent variable learning is often tackled in **stochastic computation graphs**, by estimating the gradient of an expected loss. An established method is the score function estimator (SFE) (Glynn, 1990; Williams, 1992; Kleijnen and Rubinstein, 1996). SFE is widely used in NLP, for tasks including minimum risk training in NMT (Shen et al., 2016; Wu et al., 2018) and latent linguistic structure learning (Yogatama et al., 2017; Havrylov et al., 2019). In this paper, we focus on the alternative strategy of **surrogate gradients**, which allows learning in deterministic graphs with discrete, argmax-like nodes, rather than in stochastic graphs. Examples are the **straight-through estimator (STE)** (Hinton, 2012; Bengio et al., 2013) and the structured projection of intermediate gradients optimization technique (SPIGOT; Peng et al. 2018). Recent work focuses on studying and explaining STE. Yin et al. (2019) obtained a convergence result in shallow networks for the unstructured case. Cheng et al. (2018) show that STE can be interpreted as the simulation of the projected Wasserstein gradient flow. STE has also been studied in binary neural networks (Hubara et al., 2016) and in other applications (Tjandra et al., 2019). Other methods based on the surrogate gradients have been recently explored (Vlastelica et al., 2020; Meng et al., 2020).

A popular alternative is to **relax** an argmax into a continuous transform such as softmax or sparsemax (Martins and Astudillo, 2016), as seen for instance in soft attention mechanisms (Vaswani et al., 2017), or structured attention networks (Kim et al., 2017; Maillard et al., 2017; Liu and Lapata, 2018; Mensch and Blondel, 2018; Niculae et al., 2018a). In-between surrogate gradients and relaxation is **Gumbel softmax**, which uses the Gumbel-max reparametrization to sample from a categorical distribution, applying softmax either to relax the mapping or to induce surrogate gradients (Jang et al., 2017; Maddison et al., 2017). Gumbel-softmax has been successfully applied to latent linguistic structure as well (Choi et al., 2018; Maillard and Clark, 2018). For sampling from a structured variable is required, the **Perturb-and-MAP** technique (Papandreou and Yuille, 2011) has been successfully applied to sampling latent structures in NLP applications (Corro and Titov, 2019a,b).

3 Structured Prediction Preliminaries

We assume a general latent structure model involving input variables $x \in \mathcal{X}$, output variables $y \in \mathcal{Y}$, and latent discrete variables $z \in \mathcal{Z}$. We assume that $\mathcal{Z} \subseteq \{0,1\}^K$, where $K \leq |\mathcal{Z}|$ (typically, $K \ll |\mathcal{Z}|$): *i.e.*, the latent discrete variable z can be represented as a K-th dimensional binary vector. This often results from a decomposition of a structure into parts: for example, z could be a dependency tree for a sentence of L words, represented as a vector of size $K = O(L^2)$, indexed by pairs of word indices (i,j), with $z_{ij} = 1$ if arc $i \to j$ belongs to the tree, and 0 otherwise. This allows us to define the **score** of a structure as the sum of the scores of its parts. Given a vector $s \in \mathbb{R}^K$, containing scores for all possible parts, we define

$$\operatorname{score}(z) := s^\top z. \tag{1}$$

Notation. We denote by e_k the one-hot vector with all zeros except in the k^{th} coordinate. We denote the simplex by $\triangle^{|\mathcal{Z}|} := \{p \in \mathbb{R}^{|\mathcal{Z}|} \mid p \geq 0, \sum_{z\in\mathcal{Z}} p(z) = 1\}$. Given a distribution $p \in \triangle^{|\mathcal{Z}|}$, the expectation of a function $h : \mathcal{Z} \to \mathbb{R}^D$ under p is $\mathbb{E}_{z\sim p}[h(z)] := \sum_{z\in\mathcal{Z}} p(z)h(z)$. We denote the convex hull of the (finite) set $\mathcal{Z} \subseteq \mathbb{R}^K$ by $\operatorname{conv}(\mathcal{Z}) := \{\mathbb{E}_{z\sim p}[z] \mid p \in \triangle^{|\mathcal{Z}|}\}$. The euclidean projection of s onto a set $\mathcal{D}$ is $\Pi_{\mathcal{D}}(s) := \operatorname{argmin}_{d\in\mathcal{D}} \|s - d\|_2$.

Background. In the context of structured prediction, the set $\mathcal{M} := \operatorname{conv}(\mathcal{Z})$ is known as the *marginal polytope*, since any point inside it can be interpreted as some marginal distribution over parts of the structure (arcs) under some distribution over structures. There are three relevant problems that may be formulated in a structured setting:

- Maximization (MAP inference): finds a highest scoring structure, $\texttt{MAP}(s) := \operatorname*{argmax}_{z\in\mathcal{Z}} s^\top z$.
- Marginal inference: finds the (unique) marginals induced by the scores s, corresponding to the Gibbs distribution where $p(z) \propto \exp\big(\operatorname{score}(z)\big)$. The solution maximizes the entropy-regularized objective

$$\texttt{Marg}(s) := \operatorname*{argmax}_{\mu\in\mathcal{M}} s^\top \mu + \tilde{H}(\mu), \tag{2}$$

where $\tilde{H}$ is the maximum entropy among all distributions over *structures* that achieve marginals μ (Wainwright and Jordan, 2008):

$$\tilde{H}(\mu) := \max_{\substack{p\in\triangle^{|\mathcal{Z}|}\\ \mathbb{E}_p[z]=\mu}} -\sum_{z\in\mathcal{Z}} p(z)\log p(z). \tag{3}$$

- SparseMAP: finds the (unique) *sparse* marginals induced by the scores s, given by a Euclidean projection onto $\mathcal{M}$: (Niculae et al., 2018a)

$$\begin{aligned}\texttt{SparseMAP}(s) &:= \Pi_{\mathcal{M}}(s)\\ &= \operatorname*{argmax}_{\mu\in\mathcal{M}} s^\top\mu - \frac{1}{2}\|\mu\|^2.\end{aligned} \tag{4}$$

Unstructured setting. As a check, we consider the encoding of a categorical variable with K distinct choices, encoding each choice as a one-hot vector e_k and setting $\mathcal{Z} = \{e_1, \ldots, e_K\}$. In this case, $\operatorname{conv}(\mathcal{Z}) = \triangle^K$. The optimization problems above then recover some well known transformations, as described in Table 1.

	unstructured	structured
vertices	e_k	z_k
interior points	p	μ
maximization	`argmax`	`MAP`
expectation	`softmax`	`Marg`
Euclidean projection	`sparsemax`	`SparseMAP`

Table 1: Building blocks for latent structure models.

4 Latent Structure Models

Throughout, we assume a classifier parametrized by ϕ and θ, which consists of three parts:

- An **encoder function** f_ϕ which, given an input $x \in \mathcal{X}$, outputs a vector of "scores" $s \in \mathbb{R}^K$, as $s = f_\phi(x)$;
- An **argmax node** which, given these scores, outputs the highest-scoring structure:

$$\hat{z}(s) := \operatorname*{argmax}_{z\in\mathcal{Z}} s^\top z = \texttt{MAP}(s)\,; \tag{5}$$

- A **decoder function** g_θ which, given $x \in \mathcal{X}$ and $z \in \mathcal{Z}$, makes a prediction $\hat{y} \in \mathcal{Y}$ as $\hat{y} = g_\theta(x,z)$. We will sometimes write $\hat{y}(z)$ to emphasize the dependency on z. For reasons that will be clear in the sequel, we must assume that the decoder also accepts *average structures*, *i.e.*, it can also output predictions $g_\theta(x,\mu)$ where $\mu \in \operatorname{conv}(\mathcal{Z})$ is a convex combination (weighted average) of structures.

Thus, given input $x \in \mathcal{X}$, this network predicts:

$$\hat{y} = g_\theta\left(x, \overbrace{\underset{z \in \mathcal{Z}}{\operatorname{argmax}} f_\phi(x)^\top z}^{\hat{z}(s)}\right). \tag{6}$$

To train this network, we minimize a loss function $L(\hat{y}, y)$, where y denotes the target label; a common example is the negative log-likelihood loss.

The gradient *w.r.t.* the decoder parameters, $\nabla_\theta L(\hat{y}, y)$, is easy to compute using automatic differentiation on g_θ. The main challenge is to propagate gradient information **through the argmax node** into the encoder parameters. Indeed,

$$\nabla_\phi L(\hat{y}, y) = \frac{\partial f_\phi(x)}{\partial \phi} \underbrace{\frac{\partial \hat{z}(s)}{\partial s}}_{=0} \nabla_z L(\hat{y}(\hat{z}), y) = 0,$$

so no gradient will flow to the encoder. We list below the three main categories of approaches that tackle this issue.

Introducing stochasticity. Replace the argmax node by a stochastic node where z is modeled as a random variable Z parametrized by s (*e.g.*, using a Gibbs distribution). Then, instead of optimizing a deterministic loss $L(\hat{y}(\hat{z}), y)$, optimize the **expectation** of the loss under the predicted distribution:

$$\mathbb{E}_{Z \sim p(z;s)}[L(\hat{y}(Z), y)]. \tag{7}$$

The expectation ensures that the gradients are no longer null. This is sometimes referred to as *minimum risk training* (Smith and Eisner, 2006; Stoyanov et al., 2011), and typically optimized using the *score function estimator* (SFE; Glynn, 1990; Williams, 1992; Kleijnen and Rubinstein, 1996).

Relaxing the argmax. Keep the network deterministic, but relax the argmax node into a continuous function, for example replacing it with softmax or sparsemax (Martins and Astudillo, 2016). In the structured case, this gives rise to structured attention networks (Kim et al., 2017) and their SparseMAP variant (Niculae et al., 2018a). This corresponds to moving the expectation inside the loss, optimizing $L\big(\hat{y}(\underbrace{\mathbb{E}_{Z \sim p(z;s)}[Z]}_{\mu}), y\big)$.

Inventing a surrogate gradient. Keep the argmax node and perform the usual forward computation, but backpropagate a different, non-null gradient in the backward pass. This is the approach underlying straight-through estimators (Hinton, 2012; Bengio et al., 2013) and SPIGOT (Peng et al., 2018). This method introduces a mismatch between the measured objective and the optimization algorithm. In this work, we proposed a novel, principled justification for inducing surrogate gradients. In what follows, we assume that:

- We can compute the gradient

  $$\gamma(\mu) := \nabla_\mu L(\hat{y}(\mu), y), \tag{8}$$

  for any μ, usually by automatic differentiation;[1]

- We want to replace the null gradient $\nabla_s L(\hat{y}(\hat{z}), y)$ by a surrogate $\tilde{\nabla}_s L(\hat{y}(\hat{z}), y)$.

5 SPIGOT as the Approximate Optimization of a Pulled Back Loss

We next provide a novel interpretation of SPIGOT as the minimization of a "pulled back" loss. SPIGOT uses the surrogate gradient:

$$\begin{aligned}\tilde{\nabla}_s L(\hat{y}(\hat{z}), y) &= \hat{z} - \Pi_{\mathcal{M}}(\hat{z} - \eta\gamma) \\ &= \hat{z} - \mathtt{SparseMAP}(\hat{z} - \eta\gamma),\end{aligned} \tag{9}$$

highlighting that SparseMAP (Niculae et al., 2018a) computes an Euclidean projection (Eq. 4).

5.1 Intermediate Latent Loss

To begin, consider a much simpler scenario: if we had supervision for the latent variable z (*e.g.*, if the true label z was revealed to us), we could define an **intermediate loss** $\ell(\hat{z}, z)$ which would induce nonzero updates to the encoder parameters. Of course, we do not have access to this z. Instead, we consider the following alternative:

Definition 1 (Pulled-back label)**.** A guess $\mu \in \mathcal{M} = \operatorname{conv}(\mathcal{Z})$ for what the unknown $z \in \mathcal{Z}$ should be, informed by the downstream loss.

Figure 1 provides the intuition of the pulled-back label and loss. We take a moment to justify picking $\mu \in \mathcal{M}$ rather than directly in $\mathcal{Z}$. In fact, if $K = |\mathcal{Z}|$ is small, we can enumerate all possible values of z and define the guess as the latent value minimizing the downstream loss, $\mu = \operatorname{argmin}_{z \in \mathcal{Z}} L(\hat{y}(z), y)$. This is sensible, but

[1]This gradient would not exist if the decoder g_θ were defined only at vertices $z \in \mathcal{Z}$ and not mean points $\mu \in \mathcal{M}$.

intractable in the structured case. Moreover, early on in the training process, while g_θ is untrained, the maximizing vertex carries little information. Thus, for robustness and tractability, we allow for some uncertainty by picking a convex combination $\mu \in \mathcal{M}$ so as to approximately minimize

$$\mu \approx \operatorname*{argmin}_{\mu \in \mathcal{M}} L(\hat{y}(\mu), y). \tag{10}$$

For most interesting predictive models $\hat{y}(\mu)$ (*e.g.*, deep networks), this optimization problem is non-convex and lacks a closed form solution. One common strategy is the **projected gradient algorithm** (Goldstein, 1964; Levitin and Polyak, 1966), which, in addition to gradient descent, has one more step: projection of the updated point on the constraint set. It iteratively performs the following updates:

$$\mu^{(t+1)} = \Pi_{\mathcal{M}}\left(\mu^{(t)} - \eta_t \gamma(\mu^{(t)})\right), \tag{11}$$

where η_t is a step size and γ is as in Eq. 8. With a suitable choice of step sizes, the projected gradient algorithm converges to a local optimum of Eq. 10 (Bertsekas, 1999, Proposition 2.3.2). In the sequel, for simplicity we use constant η. If we initialize $\mu^{(0)} = \hat{z} = \operatorname{argmax}_{z \in \mathcal{Z}} s^\top z$, **a single iteration** of projected gradient yields the guess:

$$\mu^{(1)} = \Pi_{\mathcal{M}}\left(\hat{z} - \eta\gamma(\hat{z})\right). \tag{12}$$

Treating the induced μ as if it were the "ground truth" label of z, we may train the encoder $f_\phi(x)$ by **supervised learning**. With a **perceptron loss**,

$$\begin{aligned} \ell_{\text{Perc}}(\hat{z}(s), \mu) &= \max_{z \in \mathcal{Z}} s^\top z - s^\top \mu \\ &= s^\top \hat{z} - s^\top \mu, \end{aligned} \tag{13}$$

a single iteration yields the gradient:

$$\nabla_s \ell_{\text{Perc}}(\hat{z}, \mu^{(1)}) = \hat{z} - \mu^{(1)}, \tag{14}$$

which is precisely the SPIGOT gradient surrogate in Eq. 9. This leads to the following insight into how SPIGOT updates the encoder parameters:

SPIGOT minimizes the **perceptron loss** between z and a pulled back target computed by **one projected gradient step** on $\min_{\mu \in \mathcal{M}} L(\hat{y}(\mu), y)$ starting at $\hat{z} = \texttt{MAP}(s)$.

This construction suggests possible alternatives, the first of which uncovers a well-known algorithm.

Relaxing the $\mathcal{M}$ constraint. The constraints in Eq. 10 make the optimization problem more complicated. We relax them and define $\mu \approx \operatorname{argmin}_{\mu \in \mathbb{R}^K} L(\hat{y}(\mu), y)$. This problem still requires iteration, but the projection step can now be avoided. One iteration of gradient descent yields $\mu^{(1)} = \hat{z} - \eta\gamma$. The perceptron update then recovers a novel derivation of **straight-through** with identity (STE-I), where the backward pass acts as if $\frac{\partial \hat{z}(s)}{\partial s} \overset{!}{=} \text{Id}$ (Bengio et al., 2013),

$$\nabla_s \ell_{\text{Perc}}(\hat{z}, \mu^{(1)}) = \hat{z} - (\hat{z} - \eta\gamma) = \eta\gamma. \tag{15}$$

This leads to the following insight into straight-through and its relationship to SPIGOT:

Straight-through (STE-I) minimizes the **perceptron loss** between z and a pulled back target computed by **one gradient step** on $\min_{\mu \in \mathbb{R}^K} L(\hat{y}(\mu), y)$ starting at $\hat{z} = \texttt{MAP}(s)$.

From this intuition, we readily obtain new surrogate gradient methods, which we explore below.

6 New Surrogate Gradient Methods

Multiple gradient updates. Instead of a single projected gradient step, we could run multiple steps of Eq. 11. We would expect this to yield a better approximation of μ. This comes at a computational cost: each update involves running a forward and backward pass in the decoder g_θ with the current guess $\mu^{(t)}$, to obtain $\gamma(\mu^{(t)}) := \nabla_\mu L(\hat{y}(\mu^{(t)}), y)$.

Different initialization. The projected gradient update in Eq. 12 uses $\mu^{(0)} = \hat{z} = \operatorname{argmax}_{z \in \mathcal{Z}} s^\top z$ as the initial point. This is a sensible choice, if we believe the encoder prediction $\hat{z}$ is close enough to the optimal μ, and it is computationally convenient, because the forward pass uses $\hat{z}$, so $\gamma(\hat{z})$ is readily available in the backward pass, thus the first inner iteration comes for free. However, other initializations are possible, for example $\mu^{(0)} = \texttt{Marg}(s)$ or $\mu^{(0)} = 0$, at the cost of an extra computation of $\gamma(\mu^{(0)})$. In this work, we do not consider alternate initializations for their own sake; they are needed for the following two directions.

Different intermediate loss: SPIGOT-CE. For simplicity, consider the unstructured case where $\mathcal{M} = \triangle$, and use the initial guess $\mu^{(0)} =$

$\texttt{softmax}(s)$. Replacing ℓ_{Perc} by the cross-entropy loss $\ell_{\text{CE}}(\mu^{(0)}, \mu^{(1)}) = -\sum_{k=1}^{K} \mu_k \log \mu_k^{(0)}$ yields

$$\nabla_s \ell_{\text{CE}}(\mu^{(0)}, \mu^{(1)}) = \mu^{(0)} - \Pi_{\triangle}(\mu^{(0)} - \eta\gamma). \quad (16)$$

In the structured case, the corresponding loss is the CRF loss (Lafferty et al., 2001), which corresponds to the KL divergence between two distributions over structures. In this case, we initialize $\mu^{(0)} = \texttt{Marg}(s)$ and update

$$\nabla_s \ell_{\text{CE}}(\mu^{(0)}, \mu^{(1)}) = \mu^{(0)} - \Pi_{\mathcal{M}}(\mu^{(0)} - \eta\gamma). \quad (17)$$

Exponentiated gradient updates: SPIGOT-EG. In the unstructured case, optimization over $\mathcal{M} = \triangle$ can also be tackled via the exponentiated gradient (EG) algorithm (Kivinen and Warmuth, 1997), which minimizes Eq. 10 with the following multiplicative update:

$$\mu^{(t+1)} \propto \mu^{(t)} \odot \exp(-\eta_t \nabla_\mu L(\hat{y}(\mu^{(t)}), y)), \quad (18)$$

where $\odot$ is elementwise multiplication and thus each iterate $\mu^{(t)}$ is strictly positive, and normalized to be inside $\triangle$. EG cannot be initialized on the boundary of $\triangle$, so again we must take $\mu^{(0)} = \texttt{softmax}(s)$. A single iteration of EG yields:

$$\begin{aligned} \mu^{(1)} &\propto \mu^{(0)} \odot \exp(-\eta\gamma) \\ &= \texttt{softmax}(\log \mu^{(0)} - \eta\gamma) \\ &= \texttt{softmax}(s - \eta\gamma). \end{aligned} \quad (19)$$

It is natural to use the cross-entropy loss, giving

$$\nabla_s \ell_{\text{CE}}(\mu^{(0)}, \mu^{(1)}) = \mu^{(0)} - \texttt{softmax}(s - \eta\gamma), \quad (20)$$

i.e., the surrogate gradient is the difference between the softmax prediction and a "perturbed" softmax. To generalize to the structured case, we observe that both EG and projected gradient are instances of mirror descent under KL divergences (Beck and Teboulle, 2003). Unlike the unstructured case, we must iteratively keep track of both *perturbed scores* and *marginals*, since $\texttt{Marg}^{-1}$ is non-trivial. This leads to the following mirror descent algorithm:

$$\begin{aligned} s^{(0)} &= s, \quad \mu^{(0)} = \texttt{Marg}(s^{(0)}), \\ s^{(t+1)} &= s^{(t)} - \eta\gamma(\mu^{(t)}), \\ \mu^{(t+1)} &= \texttt{Marg}(s^{(t)}). \end{aligned} \quad (21)$$

With a single iteration and the CRF loss, we get

$$\nabla_s \ell_{\text{CE}} = \texttt{Marg}(s) - \texttt{Marg}(s - \eta\gamma). \quad (22)$$

Algorithm 1: Surrogate gradients pseudocode: common forward pass, specialized backward passes.

```
Parameters: step size $\eta$, n. iterations $k$
Function Forward($s$, $x$, $y$):
    return $\hat{z} \leftarrow$ MAP($s$)                     // Eq. (5)
Function GradLoss($\mu, x, y$):
    return $\gamma \leftarrow \nabla_\mu L(\hat{y}(\mu), y)$          // Eq. (8)
Function BackwardSPIGOT($s, x, y$):
    $\mu^{(0)}$ = MAP($s$)
    for $t \leftarrow 1$ to $k$ do
        $\gamma \leftarrow$ GradLoss($\mu^{(t-1)}, x, y$)
        $\mu^{(t)} \leftarrow \Pi_{\mathcal{M}}(\mu^{(t-1)} - \eta\gamma)$         // Eq. (11)
    return $\mu^{(0)} - \mu^{(k)}$                          // Eq. (14)
Function BackwardSTE-I($s, x, y$):
    $\mu^{(0)}$ = MAP($s$)                                 // Eq. (15)
    for $t \leftarrow 1$ to $k$ do
        $\gamma \leftarrow$ GradLoss($\mu^{(t-1)}, x, y$)
        $\mu^{(t)} \leftarrow \mu^{(t-1)} - \eta\gamma$
    return $\mu^{(0)} - \mu^{(k)}$
Function BackwardSPIGOT-CE($s, x, y$):
    $\mu^{(0)} \leftarrow$ Marg($s$)                           // Eq. (17)
    for $t \leftarrow 1$ to $k$ do
        $\gamma \leftarrow$ GradLoss($\mu^{(t-1)}, x, y$)
        $\mu^{(t)} \leftarrow \Pi_{\mathcal{M}}(\mu^{(t-1)} - \eta\gamma)$
    return $\mu^{(0)} - \mu^{(k)}$
Function BackwardSPIGOT-EG($s, x, y$):
    $(s^{(0)}, \mu^{(0)}) \leftarrow (s$, Marg($s$))             // Eq. (21)
    for $t \leftarrow 1$ to $k$ do
        $\gamma \leftarrow$ GradLoss($\mu^{(t-1)}, x, y$)
        $s^{(t)} \leftarrow s^{(t-1)} - \eta\gamma$
        $\mu^{(t)} \leftarrow$ Marg($s^{(t)}$)
    return $\mu^{(0)} - \mu^{(k)}$
```

Algorithm 1 sketches the implementation of the proposed surrogate gradients for the structured case. The forward pass is the same for all variants: given the scores s for the parts of the structure, it calculates the MAP structure z. The surrogate gradients are implemented as custom backward passes. The function GradLoss uses automatic differentiation to compute $\gamma(\mu)$ at the current guess μ; each call involves thus a forward and backward pass through g_θ. Due to convenient initialization, the first iteration of STE-I and SPIGOT come for free, since both $\mu^{(0)}$ and $\gamma(\mu^{(0)})$ are available as a byproduct when computing the forward and, respectively, backward pass through g_θ in order to update θ. For SPIGOT-CE and SPIGOT-EG, even with $k = 1$ we need a second call to the decoder, since $\mu^{(0)} \neq \hat{z}$, so an additional decoder call is necessary for obtaining the gradient of the loss with respect to $\mu^{(0)}$. The unstructured case is essentially identical, with Marg replaced by softmax.

Model	(3 clusters) Accuracy	V-measure	(10 clusters) Accuracy	V-measure
Baselines				
Linear model	68.05±0.09	0.00±0.00	60.00±0.06	0.00±0.00
Gold cluster labels	92.40±0.06	100.00±0.00	88.50±0.10	100.00±0.00
Relaxed				
Softmax	93.15±0.33	66.88±0.97	86.45±0.33	75.07±1.18
Sparsemax	92.95±0.38	71.35±16.60	83.75±1.32	76.13±3.89
*Gumbel-Softmax	94.25±3.42	100.00±6.80	80.45±0.77	89.68±1.10
Argmax				
*ST-Gumbel	93.85±3.25	100.00±6.80	81.25±0.68	91.52±1.46
*SFE	68.45±0.33	47.73±17.65	59.80±0.58	55.56±3.30
*SFE w/ baseline	94.20±0.08	100.00±0.00	84.70±0.97	96.83±0.85
STE-S	86.95±4.01	84.44±11.61	75.95±1.10	82.83±2.75
STE-I	92.60±0.23	100.00±0.00	84.50±1.43	94.48±1.35
SPIGOT	77.90±1.26	20.53±1.85	68.80±1.02	29.24±2.24
SPIGOT-CE	93.40±2.64	97.08±13.92	83.50±0.87	94.88±1.39
SPIGOT-EG	92.70±3.04	100.00±8.27	79.40±2.03	82.29±2.15

Table 2: Discrete latent variable learning on synthetic data: downstream accuracy and clustering V-measure. Median and standard error reported over four runs. We mark stochastic methods with *.

7 Experiments

Armed with a selection of surrogate gradient methods, we now proceed to an experimental comparison. For maximum control, we first study a synthetic unstructured experiment with known data generating process. This allows us to closely compare the various methods, and to identify basic failure cases. We then study the structured case of latent dependency trees for sentiment analysis and natural language inference in English. Full training details are described in Appendix A.

7.1 Categorical Latent Variables

For the unstructured case, we design a synthetic dataset from a mixture model $z \sim \text{Categorical}(1/K)$, $x \sim \text{Normal}(m_z, \sigma I)$, $y = \text{sign}(w_z^\top x + b_z)$, where m_z are randomly placed cluster centers, and w_z, b_z are parameters of a different ground truth linear model for each cluster. Given cluster labels, one could learn the optimal linear classifier separating the data in that cluster. Without knowing the cluster, a global linear model cannot fit the data well. This setup provides a test bed for discrete variable learning, since accurate clustering leads to a good fit. The architecture, following §4, is:

- **Encoder**: A linear mapping from the input to a K-dimensional score vector: $s = f_\phi(x) = W_f x + b_f$, where $\phi = (W_f, b_f) \in \mathbb{R}^{K \times \dim(\mathcal{X})} \times \mathbb{R}^K$ are parameters.
- **Latent mapping**: $\hat{z} = \rho(s)$, where ρ is argmax or a continuous relaxation such as `softmax` or `sparsemax`.
- **Decoder**: A bilinear transformation, combining the input x and the latent variable z:
$$\hat{y} = g_\theta(x, \hat{z}) = \hat{z}^\top W_g x + b_g,$$
where $\theta = (W_g, b_g) \in \mathbb{R}^{K \times \dim(\mathcal{X})} \times \mathbb{R}$ are model parameters. If $\hat{z} = e_k$, this *selects* the k^{th} linear model from the rows of W_g.

We evaluate two baselines: a linear model, and an *oracle* where $g_\theta(x, z)$ has access to the true z. In addition to the methods discussed in the previous section, we evaluate *softmax* and *sparsemax* end-to-end differentiable relaxations, and the STE-S variant which uses the `softmax` backward pass while doing `argmax` in the forward pass. We also compare *stochastic* methods, including score function estimators (with an optional moving average control variate), and the two Gumbel estimator variants (Jang et al., 2017; Maddison et al., 2017): Gumbel-Softmax with relaxed `softmax` in the forward pass, and the other using `argmax` in the style of STE (hence dubbed ST-Gumbel).

Results. We compare the discussed methods in Table 2. Knowledge of the data-generating process allows us to measure not only downstream accuracy, but also **clustering quality**, by comparing the model predictions with the known true z. We measure the latter via the V-measure (Rosenberg and Hirschberg, 2007), a clustering score independent of the cluster labels, *i.e.*, invariant to permuting the

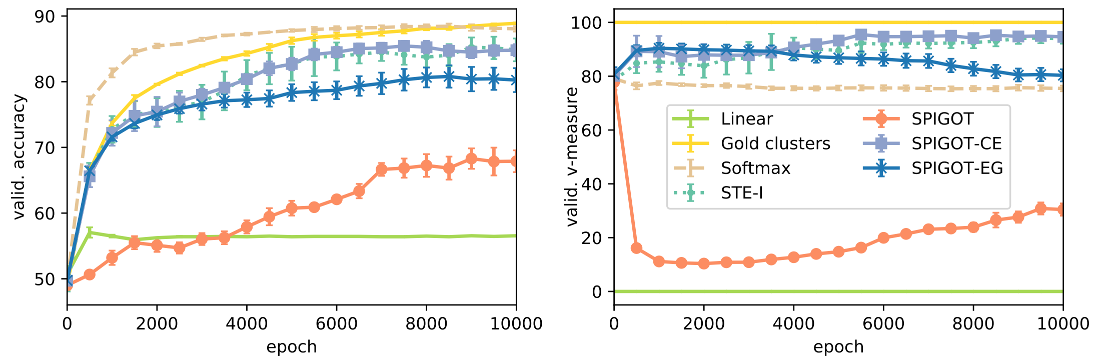

Figure 2: Learning curves on synthetic data with 10 clusters. Softmax learns the downstream task fast, but mixes the clusters, yielding poor V-measure. SPIGOT fails on both metrics; STE-I and the novel SPIGOT-CE work well.

labels (between 0 and 100, with 100 representing perfect cluster recovery). The linear and gold cluster oracle baselines confirm that cluster separation is needed for good performance. Stochastic models perform well across both criteria. Crucially, SFE requires variance reduction to performs well, but even a simple control variate will do.

Deterministic models may be preferable when likelihood assessment or sampling is not tractable. Among these, STE-I and SPIGOT-{CE,EG} are indistinguishable from the best models. Surprisingly, the vanilla SPIGOT fails, especially in cluster recovery. Finally, the relaxed deterministic models perform very well on accuracy and learn very fast (Figure 2), but appear to rely on mixing clusters, therefore they remarkably fail to recover cluster assignments.[2] This is in line with the structured results of Corro and Titov (2019b). Therefore, if latent structure recovery is less important than downstream accuracy, relaxations seem preferable.

Impact of multiple updates. One possible explanation for the failure of SPIGOT is that SPIGOT-CE and SPIGOT-EG perform more work per iteration, since they use a softmax initial guess and thus require a second pass through the decoder. We rule out this possibility in Figure 3: even when tuning the number of updates, SPIGOT does not substantially improve. We observe, however, that SPIGOT-CE improves slightly with more updates, outperforming STE-I. However, since each update step performs an additional decoder call, this also increases the training time.

[2]With relaxed methods, the V-measure is always calculated using the argmax, even though g_θ sees a continuous relaxation.

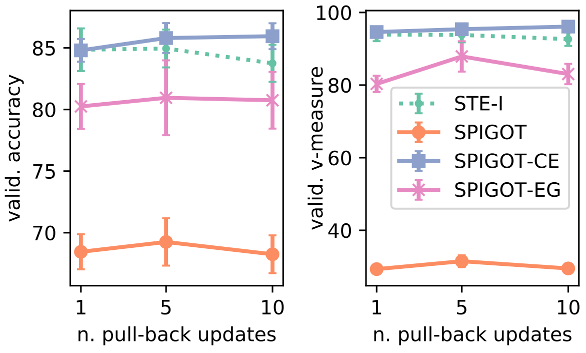

Figure 3: Impact of multiple gradient update steps for the pulled-back label, on the synthetic example with 10 clusters. For each point, the best step size η is chosen.

7.2 Structured Latent Variables

For learning structured latent variables, we study sentiment classification on the English language Stanford Sentiment Treebank (SST) (Socher et al., 2013), and Natural Language Inference on the SNLI dataset (Bowman et al., 2015).

7.2.1 Sentiment Classification

The model predicts a latent projective arc-factored dependency tree for the sentence, then uses the tree in predicting the downstream binary sentiment label. The model has the following components:

- **Encoder:** Computes a score for every possible dependency arc $i \to j$ between words i and j. Each word is represented by its embedding h_i,[3] then processed by an LSTM, yielding contextual vectors $\overleftrightarrow{h_i}$. Then, arc scores are computed as

$$s_{i\to j} = v^\top \tanh\left(W^\top[\overleftrightarrow{h_i};\overleftrightarrow{h_j}] + b\right). \quad (23)$$

[3]Pretrained GloVe vectors (Pennington et al., 2014).

Model	SST Valid. Acc.	SST Test Acc.	SNLI Valid. Acc.	SNLI Test Acc.
Baseline	83.79±0.17	83.99±0.32	85.54±0.14	85.09±0.21
Relaxed				
Marginals	84.43±0.27	83.45±0.56	85.60±0.11	85.01±0.11
SparseMAP	83.94±0.41	83.61±0.33	85.54±0.10	**85.35**±0.06
Argmax				
*Perturb-and-MAP	84.06±0.59	82.92±0.61	84.62±0.14	83.80±0.06
STE-S	83.25±0.83	83.32±0.88	82.07±0.50	81.10±0.65
STE-I	83.44±0.70	83.17±0.11	81.39±0.63	81.00±0.32
SPIGOT	84.51±0.80	**84.80**±1.10	84.03±0.28	83.52±0.24
SPIGOT-CE	82.22±0.61	83.01±0.55	80.22±1.02	79.20±0.68
SPIGOT-EG	82.94±1.06	82.88±0.90	85.36±0.16	84.84±0.16

Table 3: SST and SNLI average accuracy and standard deviation over three runs, with latent dependency trees. Baselines are described in Section 7.2. We mark stochastic methods marked with *.

- **Latent parser:** We use the arc scores vector s to get a parse $\hat{z} = \rho(s)$ for the sentence, where $\rho(s)$ is the `argmax`, or combination of trees, such as `Marg` or `SparseMAP`.
- **Decoder:** Following Peng et al. (2018), we concatenate each $\overleftrightarrow{h_i}$ with its predicted head $\overleftrightarrow{h}_{\text{head}(i)}$. For relaxed methods, we average all possible heads, weighted by the corresponding marginal: $\overleftrightarrow{h}_{\text{head}(i)} := \sum_j \mu_{i \to j} \overleftrightarrow{h_j}$. The concatenation is passed through an affine layer, a `ReLU` activation, an attention mechanism, and the result is fed into a linear output layer.

For marginal inference, we use `pytorch-struct` (Rush, 2020). For the SparseMAP projection, we use the active set algorithm (Niculae et al., 2018a). The baseline we compare our models against is a BiLSTM, followed by feeding the sum of all hidden states to a two-layer ReLU-MLP.

Results. The results from the experiments with the different methods are shown in Table 3. As in the unstructured case, the relaxed models lead to strong downstream classifiers. Unlike the unstructured case, SPIGOT is a top performer here. The effect of tuning the number of gradient update steps is not as big as in the unstructured case and did not lead to significant improvement. This can be explained by a "moving target" intuition: since the decoder g_θ is far from optimal, more accurate μ do not overall help learning.

7.2.2 Natural Language Inference

We build on top of the decomposable attention model (DA; Parikh et al., 2016). Following the setup of Corro and Titov (2019b), we induce structure on the premise and the hypothesis. For computing the score of the arc from word i to j, we concatenate the representations of the two words, as in Eq. 23. In the *decoder*, after the latent parse tree is calculated, we concatenate each word with the average of its heads. We do this separately for the premise and the hypothesis. As baseline, we use the DA model with no intra-attention.

Results. The SNLI results are shown in Table 3. Here, the straight-through (argmax) methods are outperformed by the more stable relaxation-based methods. This can be attributed to the word-level alignment in the DA model, where soft dependency relations appear better suited than hard ones.

8 Conclusions

In this work, we provide a novel motivation for straight-through estimator (STE) and SPIGOT, based on pulling back the downstream loss. We derive promising new algorithms, and novel insight into existing ones. Unstructured controlled experiments suggest that our new algorithms, which use the cross-entropy loss instead of the perceptron loss, can be more stable than SPIGOT while accurately disentangling the latent variable. Differentiable relaxation models (using softmax and sparsemax) are the easiest to optimize to high downstream accuracy, but they fail to correctly identify the latent clusters. On structured NLP experiments, relaxations (SparseMAP and Marginals) tend to overall perform better and be more stable than straight-through variants in terms of classification accuracy. However, the lack of gold-truth latent structures makes it impossible to assess recovery performance. We hope that our insights, including some of our negative results, may encourage future research on learning with latent structures.

Acknowledgments

The authors are grateful to Caio Corro and the reviewers for their valuable suggestions. This work is built on the open-source scientific Python stack (Van Rossum and Drake, 2009; Harris et al., 2020; van der Walt et al., 2011; Virtanen et al., 2020; Behnel et al., 2011; Pedregosa et al., 2011; Paszke et al., 2019). This work was supported by the European Research Council (ERC StG DeepSPIN 758969), by the P2020 project MAIA (contract 045909), and by the Fundação para a Ciência e Tecnologia through contract UIDB/50008/2020.

References

Amir Beck and Marc Teboulle. 2003. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175.

Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. 2011. Cython: the best of both worlds. *Computing in Science & Engineering*, 13(2):31–39.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *preprint arXiv:1308.3432*.

Dimitri P Bertsekas. 1999. *Nonlinear Programming*. Athena Scientific Belmont.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proc. EMNLP*. Association for Computational Linguistics.

Pengyu Cheng, Chang Liu, Chunyuan Li, Dinghan Shen, Ricardo Henao, and Lawrence Carin. 2018. Straight-through estimator as projected Wasserstein gradient flow. In *Third workshop on Bayesian Deep Learning (NeurIPS 2018)*.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *Proc. AAAI*.

Caio Corro and Ivan Titov. 2019a. Differentiable Perturb-and-Parse: Semi-supervised parsing with a structured variational autoencoder. In *Proc. ICLR*.

Caio Corro and Ivan Titov. 2019b. Learning latent trees with stochastic perturbations and differentiable dynamic programming. In *Proc. ACL*.

Jenny Rose Finkel, Christopher D Manning, and Andrew Y Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proc. EMNLP*.

Peter W Glynn. 1990. Likelihood ratio gradient estimation for stochastic systems. *Commun. ACM*, 33(10):75–84.

Alan A Goldstein. 1964. Convex programming in hilbert space. *Bulletin of the American Mathematical Society*, 70(5):709–710.

Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature*, 585:357–362.

Serhii Havrylov, Germán Kruszewski, and Armand Joulin. 2019. Cooperative learning of disjoint syntax and semantics. In *Proc. NAACL: Volume 1 (Long and Short Papers)*, pages 1118–1128, Minneapolis, Minnesota. Association for Computational Linguistics.

Geoffrey Hinton. 2012. Neural networks for machine learning. In *Coursera video lectures*.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. In *Proc. NeurIPS*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparametrization with Gumbel-Softmax. In *Proc. ICLR*.

Yoon Kim, Carl Denton, Loung Hoang, and Alexander M Rush. 2017. Structured attention networks. In *Proc. ICLR*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Proc. ICLR*.

Jyrki Kivinen and Manfred K Warmuth. 1997. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63.

Jack PC Kleijnen and Reuven Y Rubinstein. 1996. Optimization and sensitivity analysis of computer simulation models by the score function method. *European Journal of Operational Research*, 88(3):413–427.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*.

Evgeny S Levitin and Boris T Polyak. 1966. Constrained minimization methods. *USSR Computational mathematics and mathematical physics*, 6(5):1–50.

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

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. *preprint arXiv:1711.05101*.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *Proc. ICLR*.

Jean Maillard and Stephen Clark. 2018. Latent Tree Learning with Differentiable Parsers: Shift-Reduce Parsing and Chart Parsing. In *Proc. ACL*.

Jean Maillard, Stephen Clark, and Dani Yogatama. 2017. Jointly learning sentence embeddings and syntax with unsupervised tree-LSTMs. *preprint arXiv:1705.09189*.

André FT Martins and Ramón Fernandez Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proc. ICML*.

Xiangming Meng, Roman Bachmann, and Mohammad Emtiyaz Khan. 2020. Training binary neural networks using the bayesian learning rule. *Proc. ICML*.

Arthur Mensch and Mathieu Blondel. 2018. Differentiable dynamic programming for structured prediction and attention. In *Proc. ICML*.

Vlad Niculae, André FT Martins, Mathieu Blondel, and Claire Cardie. 2018a. SparseMAP: Differentiable sparse structured inference. In *Proc. ICML*.

Vlad Niculae, André FT Martins, and Claire Cardie. 2018b. Towards dynamic computation graphs via sparse latent structure. In *Proc. EMNLP*.

George Papandreou and Alan L Yuille. 2011. Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models. In *Proc. ICCV*. IEEE.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *Proc. EMNLP*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Proc. NeurIPS*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *JMLR*, 12:2825–2830.

Hao Peng, Sam Thomson, and Noah A Smith. 2018. Backpropagating through structured argmax using a SPIGOT. In *Proc. ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proc. EMNLP*.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proc. EMNLP-CoNLL*.

Alexander M. Rush. 2020. Torch-struct: Deep structured prediction library.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proc. ACL: Volume 1 (Long Papers)*, pages 1683–1692, Berlin, Germany. Association for Computational Linguistics.

David A Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proc. COLING/ACL*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. EMNLP*.

Veselin Stoyanov, Alexander Ropson, and Jason Eisner. 2011. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *Proc. AISTATS*.

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *Proc. CoNLL*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proc. ACL*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. 2019. End-to-end feedback loss in speech chain framework via straight-through estimator. In *Proc. ICASSP*. IEEE.

Kristina Nikolova Toutanova. 2005. *Effective statistical models for syntactic and semantic disambiguation*. Stanford University.

Stéfan J van der Walt, S Chris Colbert, and Gael Varoquaux. 2011. The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30.

Guido Van Rossum and Fred L. Drake. 2009. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. NeurIPS*.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J van der Walt, Matthew Brett, Joshua Wilson, K Jarrod Millman, Nikolay Mayorov, Andrew RJ Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, EA Quintero, Charles R Harris, Anne M Archibald, Antônio H Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*.

Marin Vlastelica, Anselm Paulus, Vit Musil, Georg Martius, and Michal Rolinek. 2020. Differentiation of blackbox combinatorial solvers. In *Proc. ICLR*.

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

Adina Williams, Andrew Drozdov, and Samuel R Bowman. 2018. Do latent tree learning models identify meaningful structure in sentences? *TACL*, 6:253–267.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A study of reinforcement learning for neural machine translation. In *Proc. EMNLP*, pages 3612–3621, Brussels, Belgium. Association for Computational Linguistics.

Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. 2019. Understanding straight-through estimator in training activation quantized neural nets. In *Proc. ICLR*.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *Proc. ICLR*.

A Training Details

We trained all models with AdamW optimizer (Kingma and Ba, 2014; Loshchilov and Hutter, 2018). The embeddings for the SST and SNLI experiments are initialized with Glove embeddings of size 300 (Pennington et al., 2014), available from `https://nlp.stanford.edu/projects/glove/`. The training details for all experiments are described in Table 4.

Computing Infrastructure Each experiment was run on a single GPU. The setup of the computers we used is as follows:

- GPU: Titan Xp - 12GB
  CPU: 16 x AMD Ryzen 1950X @ 3.40GHz - 128GB
- GPU: RTX 2080 Ti - 12GB
  CPU: 12 x AMD Ryzen 2920X @ 3.50GHz - 128GB

B Examples of Latent Trees

We performed a manual analysis of the trees output from the different models. We notice that, on the SST dataset, most latent trees produced by most models are flat. This agrees with related work (Williams et al., 2018; Niculae et al., 2018b). The notable exception is SPIGOT-CE, where the average tree depth on the test set is around 5 and trees seem more informative, suggesting benefits of the cross-entropy loss. Figures 4, 5, 6 show examples of the trees produced from different models.

	Synthetic Data	**SST**	**SNLI**
Data			
Where to get it	Generation script included	https://nlp.stanford.edu/sentiment/	https://nlp.stanford.edu/projects/snli/
Preprocesing	§7.1; attached code.	Neutral instances removed.	
Dataset size			
Training set	5000	6920	570K
Validation set	1000	872	10K
Test set	1000	1821	10K
Labels	2	2	3
Fixed hyperparameters			
Hidden size	100	100	200
Dropout	0	0	.2
Batch size	one batch	32	64
Number of epochs	10K	40	40
Optimized hyperparameters (maximizing validation accuracy)			
Learning rate $(\times 10^{-3})$	$\{.1, 1, 2\}$	$\{.01, .02, .05, .1, .5, 1, 2\}$	$\{.01, .1, .3, 1, 3, 10\}$ (keeping $\eta = 1$)
Pullback step size η	$\{.1, 1, 2\}$	$\{.1, 1, 10\}$	$\{.001, .01, .1, 1, 10\}$ (for best learning rate)
Number of model parameters			
Baseline	2K	150K	340K
Model with latent structure	3K	180K	420K
Runtime (minutes)			
Baseline	$<$ 1 / 1000 steps	$<$ 1 / epoch	1 / epoch
Softmax / Marginals	1	3	4
Sparsemax / SparseMAP	1	3	25
Gumbel Softmax / Perturb-and-MAP	1	5	7
STE-Softmax / STE-Marginals	1	4	6
STE-Identity	1	2	5
SPIGOT	1	3	15
SPIGOT-CE	2	4	30
SPIGOT-EG	2	5	7
Best learning rate (and pullback step size, where applicable)			
Baseline	.001	.00002	.0001
Softmax / Marginals	.002	.0001	.0001
Sparsemax / SparseMAP	.001	.00005	.0003
Gumbel Softmax / Perturb-and-MAP	.002	.00005	.0001
STE-Softmax / STE-Marginals	.002	.00005	.0003
STE-Identity	.001	.0001	.0001
SPIGOT	.002 (.1)	.0001 (.1)	.0003 (1)
SPIGOT-CE	.001 (.1)	.00005 (.1)	.0001 (.1)
SPIGOT-EG	.001 (.1)	.00005 (.1)	.0001 (.001)

Table 4: Training details and other reproducibility information.

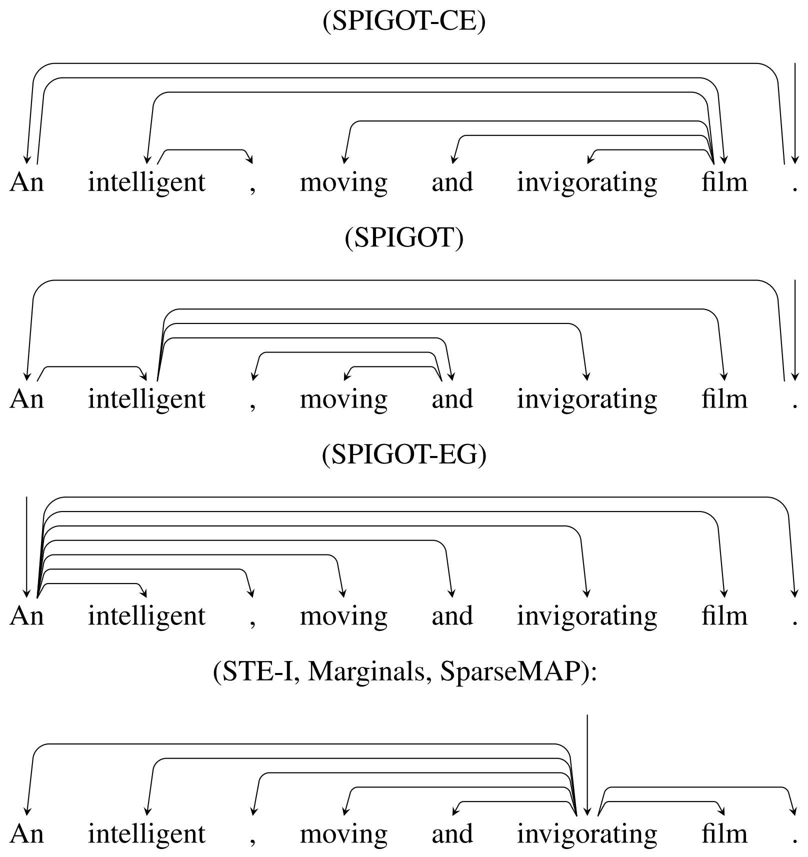
(SPIGOT-CE)
An intelligent , moving and invigorating film .
(SPIGOT)
An intelligent , moving and invigorating film .
(SPIGOT-EG)
An intelligent , moving and invigorating film .
(STE-I, Marginals, SparseMAP):
An intelligent , moving and invigorating film .

Figure 4: Example of trees.

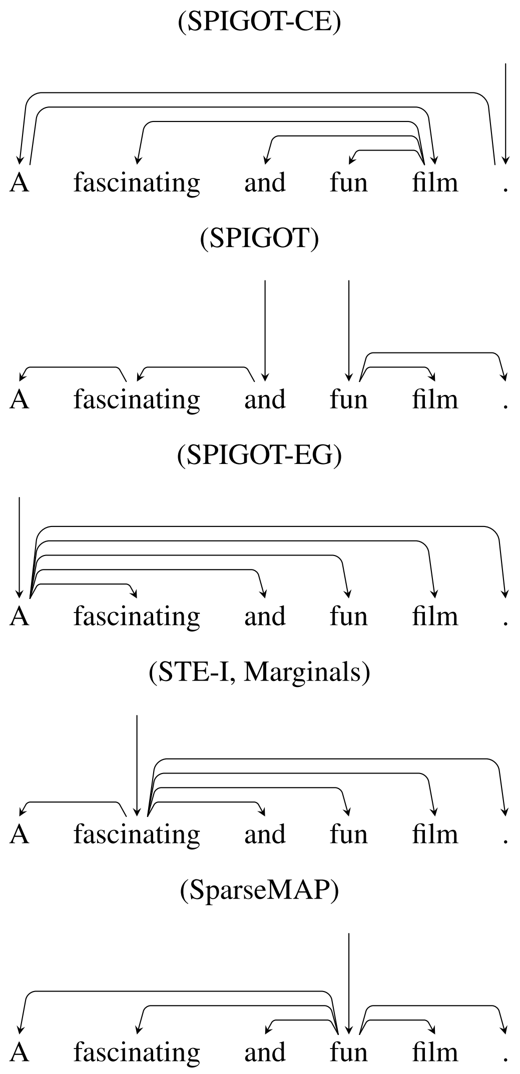
(SPIGOT-CE)
A fascinating and fun film .
(SPIGOT)
A fascinating and fun film .
(SPIGOT-EG)
A fascinating and fun film .
(STE-I, Marginals)
A fascinating and fun film .
(SparseMAP)
A fascinating and fun film .

Figure 5: Example of trees.

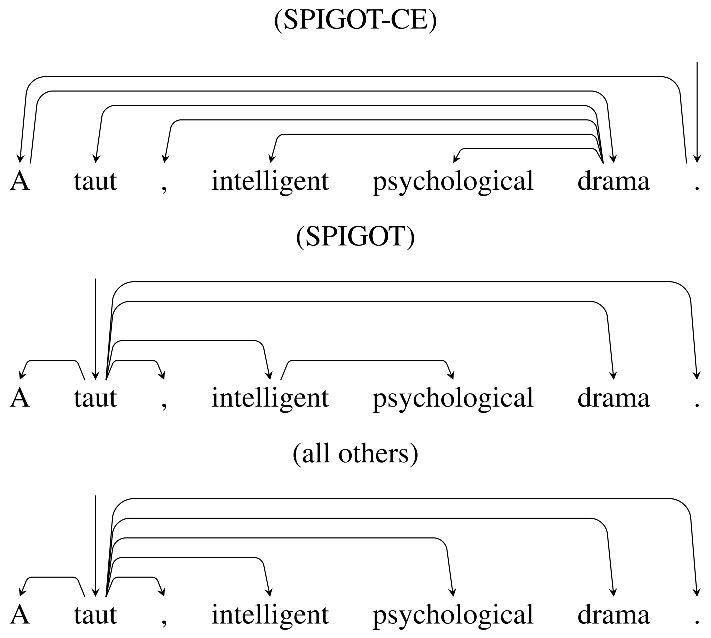
(SPIGOT-CE)
A taut , intelligent psychological drama .
(SPIGOT)
A taut , intelligent psychological drama .
(all others)
A taut , intelligent psychological drama .

Figure 6: Example of trees produced by different models for the sentence "A taut, intelligent psychological drama." The majority of the models produce mostly flat trees. In contrast, SPIGOT-CE identifies the adjectives describing the keyword "drama" and attaches them correctly.