Predict and Constrain: Modeling Cardinality in Deep Structured Prediction

Nataly Brukhim [1] **Amir Globerson** [1]

Abstract

Many machine learning problems require the prediction of multi-dimensional labels. Such structured prediction models can benefit from modeling dependencies between labels. Recently, several deep learning approaches to structured prediction have been proposed. Here we focus on capturing cardinality constraints in such models. Namely, constraining the number of non-zero labels that the model outputs. Such constraints have proven very useful in previous structured prediction approaches, but it is a challenge to introduce them into a deep learning framework. Here we show how to do this via a novel deep architecture. Our approach outperforms strong baselines, achieving state-of-the-art results on multi-label classification benchmarks.

1. Introduction

Deep structured prediction models have attracted considerable interest in recent years (Belanger et al. 2017; Zheng et al. 2015; Schwing & Urtasun 2015; Chen et al. 2015; Ma & Hovy 2016). The goal in the structured prediction setting is to predict multiple labels simultaneously while utilizing dependencies between labels to improve accuracy. Examples of such application domains include dependency parsing of text and semantic image segmentation.

Several recent approaches have proposed to combine the representational power of deep neural networks with the ability of structured prediction models to capture label dependence. This concept has been shown to be effective for various applications (Zheng et al. 2015; Schwing & Urtasun 2015; Ma & Hovy 2016; Belanger et al. 2017). However, there are still many important label dependency structures which are not fully captured by these approaches.

One of the most effective forms of label dependencies is cardinality (Tarlow et al. 2012; Tarlow et al. 2010; Milch et al. 2008; Swersky et al. 2012; Gupta et al. 2007). Namely, the fact that the overall number of labels taking a specific value has a distribution specific to the domain. For example, in natural language processing, they can express a constraint on the number of occurrences of a part-of-speech (e.g., that each sentence contains at least one verb (Ganchev et al., 2010)). In computer vision, a cardinality potential can encode a prior distribution over object sizes in an image.

Although cardinality potentials have been very effective in many structured prediction works, they have not yet been successfully integrated into deep structured prediction frameworks. This is precisely the goal of our work.

The challenge in modeling cardinality in deep learning is that cardinality is essentially a combinatorial notion, and it is thus not clear how to integrate it into a differentiable model. Our proposal to achieve this goal is based on two key observations.

First, we note that learning to predict *how many* labels are active for a given input is easier than predicting *which* labels are active. Hence, we break our inference process into two complementary components: we first estimate the label cardinality for a given input using a learned neural network, and then predict a label that satisfies this constraint.

The second observation is that constraining a label to satisfy a cardinality constraint can be approximated via projected gradient descent, where the cardinality constraint corresponds to a set of linear constraints. Thus, the overall label prediction architecture performs projected gradient descent in label space. Moreover, the above projection can be implemented via sorting, and results in an end-to-end differentiable architecture which can be directly optimized to maximize any performance measure (e.g., F_1, recall at K, etc.). Importantly, with this approach cardinality can be naturally integrated with other higher-order scores, such as the global scores considered in Belanger & McCallum 2016, and used in our model as well.

Our proposed method significantly improves prediction accuracy on several datasets, when compared to recent deep structured prediction methods. We also experiment with other approaches for modeling cardinality in deep structured prediciton, and observe that our Predict and Constrain

[1]Tel Aviv University, Blavatnik School of Computer Science. Correspondence to: Nataly Brukhim <natalybr@mail.tau.ac.il>, Amir Globerson <gamir@post.tau.ac.il>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

method outperforms these.

Taken together, our results demonstrate that deep structured prediction models can benefit from representing cardinality, and that a Predict and Constrain approach is an effective method for introducing cardinality in a differentiable end-to-end manner.

2. Preliminaries

We consider the setting of assigning L labels $\mathbf{y} = (y_1, \ldots, y_L)$ to an input $\mathbf{x} \in \mathcal{X}$. We assume y_i are binary labels, however our approach can be generalized to the multi-class case. We denote $[L] = \{1, ..., L\}$.

To model our problem in a structured prediction framework, we consider a score function $s(\mathbf{x}, \mathbf{y})$. The score is meant to reflect how "compatible" a label $\mathbf{y}$ is with an input $\mathbf{x}$. Thus, for a given input $\mathbf{x}$, the predicted label is:

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\text{argmax}}\ s(\mathbf{x}, \mathbf{y}) \tag{1}$$

In practice, we will maximize over $\mathbf{y}$ by relaxing the integrality constraint and using projected gradient ascent. We assume that $s(\mathbf{x}, \mathbf{y})$ can be decomposed as follows,

$$s(\mathbf{x}, \mathbf{y}) = \sum_i s_i(\mathbf{x}, y_i) + s_g(\mathbf{y}) + s_{z(x)}(\mathbf{y}) \tag{2}$$

where s_i is a function that depends on a single output variable (i.e., a unary potential), s_g is an arbitrary learned global potential defined over all variables and independent of the input, and $s_{z(x)}$ is a cardinality potential which constrains $\mathbf{y}$ to be of cardinality $z(x)$, as explained in Section 3. For brevity, we denote the cardinality potential as s_z.

Concretely, $s_i(\mathbf{x}, y_i)$ is defined as the multiplication of y_i by a linear function of a feature representation of the input, where the feature representation is given by a multi-layer perceptron $f(\mathbf{x})$. The global score s_g is obtained by applying a neural network over the variables, and evaluates such assignments independent of $\mathbf{x}$, similarly to the architecture used by Belanger & McCallum 2016.

The components of the score function $s(\mathbf{x}, \mathbf{y})$ are parameterized by a set of weights $\mathbf{w}$. As in Belanger & McCallum 2016, we learn these by end-to-end minimization of a loss function on the training data. In what follows we describe our method in more detail.

3. Learning with Cardinality Potentials

We next explain how cardinality is modeled in our score function, and how the score is maximized. Our cardinality score has two key components. First, a learned cardinality predictor $z = h(\mathbf{x})$ that maps an input to an estimate of the cardinality of $\mathbf{y}$. Second, the cardinality potential $s_z(\mathbf{y})$,

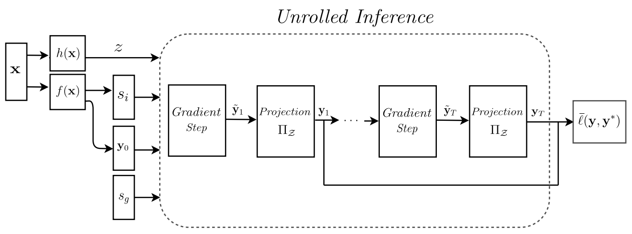

Figure 1. Deep structured model with unrolled projected gradient ascent inference.

which corresponds to the hard constraint that the cardinality of $\mathbf{y}$ is equal to z. Namely:

$$s_z(\mathbf{y}) = \begin{cases} 0 & \text{if } \sum_i y_i = z \\ -\infty & \text{otherwise} \end{cases} . \tag{3}$$

We refer to the method of using this two component potential as a Predict and Constrain approach.

3.1. Inference with Projected Gradient Ascent

We turn to the problem of maximizing the score function with respect to $\mathbf{y}$. Exact maximization is intractable for the potentials we consider. As in Belanger & McCallum 2016 we address this by first relaxing the integrality constraint on $\mathbf{y}$, and using gradient ascent for inference. However, the cardinality constraint means that we cannot use naive gradient ascent on the relaxed $\mathbf{y}$. Instead we will repeatedly project $\mathbf{y}$ onto the constrained space.

We first relax the discrete constraint on the binary variables $\mathbf{y}$ to a continuous constraint $y_i \in [0, 1]$. Next, the score $s(\mathbf{x}, \mathbf{y})$ is maximized with respect to the relaxed $\mathbf{y}$ using projected gradient ascent, as depicted in Figure 1. At each iteration of projected gradient ascent, the following two updates are performed. First, we apply a gradient update step over the score function, excluding the cardinality potential, with respect to $\mathbf{y}$. Second, we project it onto the cardinality constrained space, as explained in Section 3.2.

The above projected-gradient updates are repeated for T iterations, where the initial variables $\mathbf{y}_0$ are given by a sigmoid function applied over the unary terms. Since all update operations are differentiable, this results in an end-to-end differentiable architecture. Thus, we can directly apply a loss function to the output of this network $\mathbf{y}_T$, and optimize it (see Belanger et al. 2017; Maclaurin et al. 2015).

Let $\ell(\mathbf{y}, \mathbf{y}^*)$ be a label-loss function, reflecting the error of predicting a label $\mathbf{y}$ where the true label is $\mathbf{y}^*$. For example, $\ell(\mathbf{y}, \mathbf{y}^*)$ might be the cross-entropy loss function, or a differentiable approximation of any performance measure of interest, such as the F_1 measure we use in our experiments (see Equation (10)).

Given training data $(\mathbf{x}, \mathbf{y}^*)$, let $\mathbf{y}_t(\mathbf{x})$ be the value of label $\mathbf{y}$ after t iterations of projected gradient ascent. We would like to minimize the final loss $\ell(\mathbf{y}_T(\mathbf{x}), \mathbf{y}^*)$. As in other end-to-end approaches (Belanger et al., 2017), we found it better to apply the loss function over all T iterations. Specifically, we minimize the following objective:

$$\bar{\ell}(\mathbf{x}, \mathbf{y}^*) = \frac{1}{T}\sum_{t=1}^{T} \frac{\ell(\mathbf{y}_t(\mathbf{x}), \mathbf{y}^*)}{T - t + 1}$$

By using all iterates in the objective, the model learns to converge quickly during inference. Additionally, the fact that the loss function incorporates all layers helps avoiding vanishing gradients.

3.2. Enforcing Cardinality via Projection

We now turn to explain the implementation of the projection step. During inference we iteratively apply projected gradient ascent over the variables for T steps. In each step $t \in [T]$ of the inference pipeline we update the label variables as follows,

$$\tilde{\mathbf{y}}_{t+1} = \mathbf{y}_t + \eta \nabla_{\mathbf{y}} \bar{s}(\mathbf{x}, \mathbf{y}) \tag{4}$$

$$\mathbf{y}_{t+1} = \Pi_{\mathcal{Z}}\left[\tilde{\mathbf{y}}_{t+1}\right] \tag{5}$$

where $\bar{s}(\mathbf{x}, \mathbf{y}) = \sum_i s_i(y_i, \mathbf{x}) + s_g(\mathbf{y})$, η is the inference learning rate, and $\Pi_{\mathcal{Z}}$ denotes an approximate projection operator, described below. The gradient update step is given by Equation (4), and Equation (5) corresponds to the projection step. The operator $\Pi_{\mathcal{Z}}$ differentiably computes an approximation of a Euclidean projection onto a set defined as,

$$\mathcal{Z} = \{\mathbf{y} | \forall i. y_i \in [0, 1], \sum_i y_i = z\}$$

with z obtained using the cardinality predictor $z(\mathbf{x})$ [1]. Thus, $\Pi_{\mathcal{Z}}(\mathbf{v})$ approximately solves the following problem,

$$\begin{aligned} \underset{\mathbf{u}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{v} - \mathbf{u}\|_2^2 \\ \text{subject to} \quad & \sum_{i=1}^{L} u_i = z, \\ & 0 \le u_i \le 1, \ \forall i \in [L]. \end{aligned} \tag{6}$$

First, we note that it is possible to compute the above minimization directly by solving the convex optimization problem in (6). However, it does not naturally translate into an end-to-end differentiable network, as detailed in Section 4.1. Instead, we construct operator $\Pi_{\mathcal{Z}}$ using two sub-processes, each computing a projection onto a different set, $\mathcal{A}$ and $\mathcal{B}$, such that $\mathcal{Z} = \mathcal{A} \cap \mathcal{B}$. Let $\mathcal{A} = \{\mathbf{y} | \forall i. y_i \le 1\}$. Given $\mathbf{y}$, its Euclidean projection onto $\mathcal{A}$ is obtained simply by clipping values larger than 1, i.e. $\Pi_{\mathcal{A}}(\mathbf{y}) = \min(\mathbf{y}, 1)$. Let $\mathcal{B} = \{\mathbf{y} | \forall i. y_i \ge 0, \sum_i y_i = z\}$, the positive simplex. When $z = 1$, $\mathcal{B}$ is the probability simplex. The Euclidean projection onto $\mathcal{B}$ can be done using an $O(L \log L)$ algorithm which relies on sorting (Duchi et al., 2008).

[1]Note that $z(\mathbf{x})$ need not be integral as the variables themselves were relaxed to the $[0, 1]$ range.

Algorithm 1 Soft projection onto the simplex

```
**Input:** vector $\mathbf{v} \in \mathbb{R}^L$, cardinality $z \in [Z]$
Sort $\mathbf{v}$ into $\boldsymbol{\mu}$: $\mu_1 \ge ... \ge \mu_L$
Compute $\bar{\boldsymbol{\mu}}$ the cumulative sum of $\boldsymbol{\mu}$
Let $\mathbf{I}$ be a vector of indices $1, ..., L$.
$\boldsymbol{\delta} = \text{Softsign}\big(\boldsymbol{\mu} \circ \mathbf{I} - (\bar{\boldsymbol{\mu}} - z)\big)$
$\boldsymbol{\rho} = \text{Softmax}\big(\boldsymbol{\delta} \circ \mathbf{I}\big)$
$\theta = \frac{1}{\mathbf{I}^{\mathrm{T}}\boldsymbol{\rho}}\big(\bar{\boldsymbol{\mu}}^{\mathrm{T}}\boldsymbol{\rho} - z\big)$
**Output:** $\max(\mathbf{v} - \theta, 0)$
```

In designing our end-to-end differentiable network, we must take into account the smoothness restriction of our network components. Hence, we devise a differentiable variation of the simplex projection algorithm. The soft procedure for computing $\Pi_{\mathcal{B}}(\mathbf{y})$ is given in Algorithm 1. For differential sorting we use a sorting component based on a differential variation of Radix sort, built in the deep learning library TensorFlow (Abadi et al., 2015). It is an interesting direction for future research to explore other differentiable sorting operations that can be computed more efficiently.

Next, we need to combine the outputs of the operators $\Pi_{\mathcal{A}}(\mathbf{y})$ and $\Pi_{\mathcal{B}}(\mathbf{y})$ to output the desired $\Pi_{\mathcal{Z}}(\mathbf{y})$. If $\mathcal{A}$ and $\mathcal{B}$ were affine sets we could have applied the alternating projection method (Escalante & Raydan, 2011), by alternately projecting onto the sets $\mathcal{A}$ and $\mathcal{B}$. In this case, the method is guaranteed to converge to the Euclidean projection of $\mathbf{y}$ onto the intersection $\mathcal{Z} = \mathcal{A} \cap \mathcal{B}$. Since these are not affine sets due to the inequality constraints, this method is only guaranteed to converge to some point in the intersection.

Instead, we use Dykstra's algorithm (Boyle & Dykstra, 1986) which is a variant of the alternating projection method. Dykstra's algorithm converges to the Euclidean projection onto the intersection of convex sets, such as $\mathcal{A}$ and $\mathcal{B}$. Specifically, for each step $r \in [R]$ for a fixed number of iterations R we compute the following sequence,

$$\begin{aligned} \tilde{\mathbf{y}}^{(r)} &= \Pi_{\mathcal{A}}(\mathbf{y}^{(r)} + \mathbf{p}^{(r)}) \\ \mathbf{p}^{(r+1)} &= \mathbf{y}^{(r)} + \mathbf{p}^{(r)} - \tilde{\mathbf{y}}^{(r)} \\ \mathbf{y}^{(r+1)} &= \Pi_{\mathcal{B}}(\tilde{\mathbf{y}}^{(r)} + \mathbf{q}^{(r)}) \\ \mathbf{q}^{(r+1)} &= \tilde{\mathbf{y}}^{(r)} + \mathbf{q}^{(r)} - \mathbf{y}^{(r+1)} \end{aligned}$$

Where $\mathbf{p}^{(0)} = \mathbf{q}^{(0)} = 0$. Empirically, we find that a small number of iterations is sufficient, and we set $R = 2$ in all our experiments.

4. Review of Alternative Approaches

In this section we review alternative approaches to modeling cardinality. Our goal is to further examine possible directions of dealing with complex global structures of the output labels, as well as to demonstrate the motivation behind the design choices made in our deep structured architecture.

First, we consider an architecture which only consists of the unary and cardinality scores, s_i and s_z, discarding the global potential s_g. In this case, exact maximization is possible by simply sorting the unary terms and obtaining the top z values. The maximizer $\mathbf{y}^*$ is the binary vector in which the labels corresponding to the top z values are on. This approach can be trained using standard structured hinge loss.

Although appealing in its simplicity, this method fails to perform as well without utilizing the expressiveness of the global score s_g, which captures variable interactions that cannot be modeled by cardinality or unary potentials alone. SPENs (Belanger & McCallum, 2016) have demonstrated the expressive power of such global scores to captures important structural dependencies among labels, such as mutual exclusivity and implicature.

The $\mathbf{y}_i \in [0, 1]$ constraint may be achieved directly by setting $\mathbf{y} = \sigma(\boldsymbol{\alpha})$ where σ is the sigmoid function. This is the approach taken in Belanger et al. 2017. We could have used this representation instead of clipping, though due to our implementation of cardinality constraints as projection in $\mathbf{y}$ space, clipping is more natural.

Since the global score s_g is used in our framework, it can in principle model cardinality, eliminating the need for the cardinality potential s_z. However, such a neural network would require using a deeper structure with more parameters, and is thus prone to overfitting. Additionally, the projection operator used in our network is tailored for the cardinality case, and is thus more effective. We have experimented with an architecture which relies only on unary and global scores, and observed that adding cardinality to those improves performance, as demonstrated in Section 6.

Our cardinality component introduces a hard constraint on label counts. Alternatively, the projection component of our pipeline could conceivably be replaced with a cardinality-dependent potential as follows. Let w_z be a score assigned to labels $\mathbf{y}$ with $\sum_i y_i = z$. To implement this in a differentiable manner, define $I_z(\mathbf{y}) = \sigma\big(\sum_i y_i - z\big)$, where σ is the sigmoid function. We then define,

$$s_z(\mathbf{y}) = w_z \cdot I_z(\mathbf{y}) \cdot \big(1 - I_{z+1}(\mathbf{y})\big) \tag{7}$$

If a threshold function was used instead of the sigmoid function, this term exactly assigns a score w_z to labels of cardinality z. By using the sigmoid function we obtain a differentiable representation of that potential. The advantage of that approach is that no projection is needed and a simple optimization over $y_i \in [0, 1]$ is sufficient. However, in practice we have found that it does not perform as well as our Predict and Constrain approach, as shown in Section 6.

The potential defined in Equation (7) is another way of modeling "hard" cardinality in a differentiable way. Instead of choosing this "hard" modeling form, we have also examined a "softer" notion of cardinality. Specifically, setting $s_z(\mathbf{y}) = w_z \cdot \|\sum_i y_i - z\|_2^2$, encourages the label count to be within a small range from a predicted (or fixed) cardinality z.

However, similarly to the s_z discussed above, these potentials have a smaller effect on the inner optimization than applying projection, while a natural way of modeling cardinality via projection is in a "hard" form. Essentially, the design of s_z as in Equation (3), and its differentiable implementation as a projection operation, exhibits the best empirical performance compared to the alternatives discussed above.

Finally, it is possible to frame the task of maximizing $s(\mathbf{x}, \mathbf{y})$ as a mixed integer linear program, obtaining an exact inference scheme for our network.[2] In order to train our model we could use the standard structured hinge loss. This formulation could also be relaxed to a linear program making it more efficient to solve. However, solving an LP for each prediction is impractical.

4.1. Fast Exact Projection

Our projection scheme relies on Dykstra's algorithm which requires us to alternately apply $\Pi_{\mathcal{A}}$ and $\Pi_{\mathcal{B}}$ in an iterative fashion. In order to solve optimization problem 6 to optimality we would need to encode several iterates of these alternations in our computation graph, to form a deeper network.

Instead, we could compute the projection $\Pi_{\mathcal{Z}}$ by solving optimization problem 6 directly, without separately applying $\Pi_{\mathcal{A}}$ and $\Pi_{\mathcal{B}}$. A solution to problem in Equation 6 was given by Gupta et al. 2010. They also describe a fast linear-time algorithm to obtain the maximizer $\mathbf{u}^*$. We will give a brief description of their method.

Assume w.l.o.g. that $\mathbf{v}$ is sorted such that $v_1 \geq ... \geq v_n$. Let ρ_1, ρ_2 be the indices up until which all projected values are ones, and after which all projected values are zeros, respectively. Then, the values of the maximizer $\mathbf{u}^*$ take the

[2]Assuming the global potential s_g is designed as a multi-layer neural network with ReLU activations.

following form,

$$u_i = \begin{cases} 0 & \text{if } v_i - \lambda \le 0 \\ v_i - \lambda & \text{if } 0 < v_i - \lambda < 1 \\ 1 & \text{if } v_i - \lambda \ge 1 \end{cases} \tag{8}$$

with λ defined as follows,

$$\lambda = \frac{\sum_{i=\rho_1+1}^{\rho_2} v_i - (z - \rho_1)}{\rho_2 - \rho_1} \tag{9}$$

The algorithm suggested in Gupta et al. 2010 computes λ in an iterative fashion, based on the fact that z is a piece-wise linear function in λ with points of discontinuity at values $\mathbf{v}$ and $\tilde{\mathbf{v}} = \max(\mathbf{v} - 1, 0)$. The algorithm requires maintaining an uncertainty interval for λ, which is initialized at $[\min(\tilde{\mathbf{v}}), \max(\mathbf{v})]$. In each iteration $j \in [J]$ we obtain λ_j, the median of the merged set of unique values of $\tilde{\mathbf{v}}$ and $\mathbf{v}$, lying in the current uncertainty interval. We then need to compare z_j to z, where z_j can be computed using Equation 9. The size of the uncertainty interval is reduced in every iteration, until the correct value is recovered.

Although this method efficiently computes the correct projection, it requires applying non-trivial combinatorial operations which do not naturally translate to differentiable operations, such as set-union, and median.

We have experimented with a differentiable implementation of this algorithm in Section 6. The projection algorithm requires many components added to the computation graph making it deeper and thus harder to backpropagate through. Alternatively, by iterating for only a small number of iterations, the resulting λ_J is far from the correct λ.

Overall, in this end-to-end setting, we found it to have inferior performance compared to the alternating projection method described in Section 3.2. Instead, the use of Dykstra's algorithm with a few alternating projection iterations, was both efficient and thus easier to differentiate through, and resulted in a good approximation of the correct maximizer.

5. Related Work

Several recent approaches have applied gradient-based inference to a variety of structured prediction tasks (Belanger & McCallum 2016; Gygli et al. 2017; Amos & Kolter 2017). Specifically, Structured Prediction Energy Networks (SPENs) (Belanger & McCallum, 2016) optimize the sum of local unary potentials and a global potential and are trained with a structured SVM loss. Another approach is the Deep Value Network (DVN) (Gygli et al., 2017) which uses an energy network architecture similar to SPEN, but instead trains it to fit the task cost function.

Our architecture was constructed similarly to SPEN and DVN for the unary and global potentials s_i, and s_g, though we extended the expressivity of this architecture both by introducing the cardinality potential s_z, as well as an effective inference method for the overall score.

Input Convex Neural Networks (ICNNs) (Amos & Kolter, 2017) design potentials which are convex with respect to the labels, so that inference optimization will be able to reach global optimum. Their design achieves convexity by restricting the model parameters and activation functions, limiting the expressiveness of the learned potentials. In practice, it has inferior performance compared to its non-convex counterpart, as shown by Belanger et al. 2017.

Our approach differs from these methods mainly in its capability of encapsulating complex global dependencies in the form of cardinality potentials, which are harder to obtain using general form global potentials, optimized with an effective end-to-end inference method.

5.1. Cardinality Potentials

The use of higher-order global potentials, and specifically of cardinality relations, have shown to be useful in a wide range of applications. For example, in computer vision they have been used to improve human activity recognition (Hajimirsadeghi et al., 2015) by considering the number of people involved in an activity, which is harder to infer using spatial relations alone. In part-of-speech tagging, cardinalities can enforce the constraint that each sentence must contain at least one verb (Ganchev et al., 2010).

The properties of cardinality potentials and corresponding inference methods have been explored previously (Tarlow et al. 2012; Tarlow et al. 2010; Milch et al. 2008; Swersky et al. 2012; Gupta et al. 2007). General form global potentials often result in non-trivial dependencies between variables that make exact inference intractable, thus requiring the use of approximate inference methods.

Conversely, MAP inference for cardinality potential models is well-understood. Notably, Gupta et al. 2007 shows an exact MAP inference algorithm, and Tarlow et al. 2010 gives a an algorithm for computing the cardinality potential messages for max-product belief propagation, both algorithms are solved efficiently in $O(L\log L)$ time. Still, when considering general form global potentials as in our framework, we must employ an approximate inference scheme which can be computed efficiently.

5.2. Unrolled Optimization

End-to-end training with unrolled optimization was first used in deep networks by Maclaurin et al. 2015 for tuning hyperparameters. More recently, other approaches have unrolled gradient-based methods within deep networks in various different contexts (Metz et al. 2016; Andrychowicz et al. 2016; Greff et al. 2017).

In the context of inference, Belanger et al. 2017 explored SPENs which use gradient descent to approximate energy minimization, while learning the energy function end-to-end. In computer vision, several works have incorporated structured prediction methods like conditional random fields within neural networks (Zheng et al. 2015; Schwing & Urtasun 2015), and the Mean-Field algorithm was used for inference. Recent work by Larsson et al. 2017 has used projected gradient descent for CRF inference with pairwise potentials, where they project the variables onto the simplex, rather than constraining their values based on learned potentials via projection.

An important advantage of these training schemes is that they return not only the learned potentials, but also an actual inference optimization method, tuned on the training data, to be used at test time. However, these methods are either restricted to basic graphical models (e.g with pairwise or low order clique potentials) to ensure tractability, or have used global potentials implemented via multi-layer fully connected nets. Our approach harnesses the effectiveness of unrolled optimization while boosting its ability to infer important structures expressed by cardinality potentials.

6. Experiments

We evaluate our method on multi-label document classification (MLC). The MLC task is relevant in a wide range of machine learning applications, and characterized by higher-order labels interaction, which can be addressed by our deep structured network. Therefore, it is a natural application of our method. We use 3 standard MLC benchmarks, as used by other recent approaches (Belanger & McCallum 2016; Gygli et al. 2017; Amos & Kolter 2017): Bibtex, Delicious, and Bookmarks.

6.1. Cardinality Prediction Analysis

We begin by demonstrating the effectiveness of estimating the cardinality of a label $\mathbf{y}$ given the input $\mathbf{x}$. We train a simple feed-forward neural network which consists of a single hidden layer with ReLU activations, with the goal of predicting the ground-truth cardinality. The output layer is a softmax over Z output neurons, where Z is allowed the maximal cardinality of the labels. This is the same architecture we used for $h(\mathbf{x})$ in the MLC experiments below. For the current experiment, we use the correct cardinality $|\mathbf{y}^*|$ as ground truth and optimize $h(\mathbf{x})$ to maximize prediction accuracy.

We evaluate the results on the Delicious dataset using the mean squared error of our predictor output with respect to the correct cardinality. We compare our predictor to a random baseline over the range of $[0, 25]$, which is the range of possible cardinalities in the data, as well as to the constant cardinality of 19 which is the average cardinality in the training data. Our predictor performs better than both baselines, with $\text{MSE}_{\text{rand}} = 74.9$, $\text{MSE}_{\text{const}} = 26.9$, and $\text{MSE}_h = 19.5$.

A possible explanation for this phenomenon is that some attributes of the input data might indicate approximately how many active labels the label set contains. Features such as the number of distinct words, or the existence or absence of specific meaningful words could be relevant here, making the task of inferring *the number of active labels* easier than predicting *which* labels are active. For example, an article with many distinct words suggests that it discusses a broad range of subjects, and thus relates to many different tags, while an article with few distinct words is more likely to be focused on a specific subject and therefore has only a small set of tags. Learning which combination of input words corresponds to which specific tagging set is harder than learning to predict cardinality based on feature representations of simpler forms, such as the ones discussed above.

The other datasets we tested, Bibtex and Bookmarks, are extremely sparse with average cardinalities of 2.4 and 2, respectively. The task of predicting the correct cardinality within this smaller possible range is harder. Accordingly, the predictor $h(\mathbf{x})$ approximately learns the average cardinality, and we observed in our experiments that projecting to a predicted cardinality $z = h(\mathbf{x})$ yields similar performance to projection onto a set defined by constant cardinality z. However, using the predictor $h(\mathbf{x})$ did manage to improve our overall performance for the Bibtex dataset compared to a fixed z, whereas for Bookmarks our results are slightly lower than for using a fixed z.

The Predict and Constrain approach of first estimating the relevant cardinality and then projecting onto the cardinality-constrained space is especially useful for datasets of larger average cardinality and cardinality variance, such as Delicious, for which we obtained a significant performance improvement using this method.

6.2. Experimental Setup

The evaluation metric for the MLC task is the example averaged (macro-average) F_1 measure. We found it useful to use its continuous extension as our loss function $\ell(\mathbf{y}, \mathbf{y}^*)$, similarly to Gygli et al. 2017, i.e.,

$$\ell(\mathbf{y}, \mathbf{y}^*) = -\frac{2\mathbf{y}^{\mathrm{T}}\mathbf{y}^*}{\sum_i (y_i + y_i^*)} \tag{10}$$

where $\mathbf{y}^*$ is binary and $\mathbf{y} \in [0, 1]$. We observed improved performance by training with $\ell(\mathbf{y}, \mathbf{y}^*)$ as opposed to alternative loss functions such as cross entropy.

Table 1. F_1 performance of our approach compared to the state-of-the-art on multi-label classification.

DATASET	BIBTEX	BOOKMARKS	DELICIOUS
SPEN	42.2	34.4	37.5
E2E-SPEN	38.3	33.9	35.1
DVN	44.7	37.1	-
MLP	38.9	33.8	37.8
SC	42.0	34.6	34.6
FP	42.1	36.0	34.2
NC	40.3	35.9	36.3
OURS	**45.5**	**39.1**	**38.4**

The architecture used for all datasets consists of neural networks for the unary potentials s_i, the global potential s_g, and the cardinality estimator $h(\mathbf{x})$, respectively. For all neural networks we use a single hidden layer, with ReLU activations. For the unrolled optimization we used gradient ascent with momentum 0.9, unrolled for T iterations, with T ranging between $10-20$, and with $R=2$ alternating projection iterations. All of the hyperparameters were tuned on development data. We trained our network using AdaGrad (Duchi et al., 2011) with learning rate $\eta = 0.1$.

We compare our method to the following baselines:

- SPEN - Structured Prediction Energy Networks (Belanger & McCallum, 2016) use gradient-based inference to optimize an energy network of local and global potentials, and are trained with a structured-SVM loss.
- E2E-SPEN - an end-to-end version of SPEN (Belanger et al., 2017).
- DVN - Deep Value Networks (Gygli et al., 2017) train an energy function to estimate the task loss on different labels for a given input, with gradient-based inference.
- MLP - a multi-layer perceptron with ReLU activations trained with a cross-entropy loss function.

The MLP and SPEN baseline results were taken from (Belanger & McCallum, 2016). The E2E-SPEN results were obtained by running their publicly available code on these datasets. In our experiments we follow the same train and test split as the baseline methods on all datasets. The results are shown in Table 1.

6.3. Alternative Implementations

In addition to prior work, we also compared our approach to alternative implementation of cardinality constraints, as discussed in Section 4. The goal of this comparison is to examine the role of our unrolled projection scheme in capturing cardinality structure. Specifically, we consider the following methods,

- SC - The sigmoid-based indicators as cardinality potentials, discussed in Section 4.
- FP - The fast exact projection method described in Section 4.1.
- NC - A simple baseline which uses our architecture excluding cardinality potentials altogether.

All methods used unrolled gradient ascent inference, and the negative F_1 loss function in Equation (10).

The fast projection method was implemented using a differentiable approximation of the projection algorithm steps. The algorithm performs a binary search over the values of z to obtain the correct λ, using set-union and median operations. Instead, we maintain a lower and upper bound for the uncertainty interval, and in each iteration we compute the average, rather than the median, until the gap between the lower and upper bounds is below a threshold. In each iteration j, we compared z to z_j, given by Equation (9).

To obtain the values ρ_1, ρ_2, we compute $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_2$, the one-hot encodings of ρ_1, ρ_2 at every iteration j. Let $\boldsymbol{\mu}$ be the sorted values of the vector we wish to project. We first compute $\boldsymbol{\delta}_1^{(j)} = \text{Softsign}(\boldsymbol{\mu} - \lambda^{(j)} - 1)$ and $\boldsymbol{\delta}_2^{(j)} = \text{Softsign}(\boldsymbol{\mu} - \lambda^{(j)})$. Then, we apply Softmax over the element-wise multiplication of a fixed indices vector and $\boldsymbol{\delta}_1^{(j)}$ or $\boldsymbol{\delta}_2^{(j)}$, to obtain $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_2$, respectively. We compute the dot-product of the one-hot encodings and the cumulative sum vector of $\boldsymbol{\mu}$ to obtain the partial sum in Equation (9). Finally, we use Equation (8) to obtain the projected values.

6.4. Results

It can be seen from Table 1 that our method outperforms all baselines we have compared to, obtaining state-of-the-art results in these tasks. Comparing our network to another end-to-end gradient-based inference method, the E2E-SPEN achieves significantly low performance. As stated by the authors in their release notes, their method is prone to overfitting and actually performs worse than the original SPEN on these benchmarks. Additionally, our method improves upon the original SPEN by a large margin.

Our method also outperformed DVN on the Bibtex and Bookmarks datasets. They DVN paper did not report results on the Delicious dataset, and running the public DVN code yielded low accuracy on it, which suggests that further fine tuning of their method is required for different datasets. The Delicious dataset has the largest label space with 982 labels, while Bibtex and Bookmarks have 159 and 208, respectively, and is therefore the most challenging. Thus, these results illustrate the robustness of our method, as it achieves superior performance for Delicious over all baselines, albeit not being specifically tuned for it.

The performance of the NC, SC, and FP baselines is surpassed by that of our network. Nevertheless, they obtain competitive results compared to strong baselines. This further demonstrates the effectiveness of unrolled optimization over applying an inference scheme separately from the training process.

Moreover, the results obtained by the SC and FP baselines indicate the power of combining general form global potentials with cardinality-based potentials to represent complex structural relations. The role of cardinality modeling in our framework is also evident from the inferior results obtained by the NC baseline, which does not utilize cardinality information.

Note that we have used the same unary and non-cardinality global scores for all methods compared. This highlights the fact that improvement in performance is due to the use of cardinality potentials, coupled with our Predict and Constrain approach.

7. Extension to Non-Binary Labels

Thus far we considered binary labels y_i. In this section we discuss extensions to the case where each y_i can have more than two values. For example, in image segmentation tasks in which each pixel could be assigned to one of multiple classes, cardinality potentials can enforce constraints on the size of labeled objects and encourage smoothness over large groups of pixels.

To re-frame our method in the general non-binary form, consider a matrix $\mathbf{Y} \in \{0,1\}^{L\times M}$ where M is the number of possible classes, and L the number of labels. Here, every row corresponds to the one-hot representation of the class $j \in [M]$ which label $i \in [L]$ is assigned to.

To facilitate continuous optimization, we relax the matrix values to lie in the continuous interval $\mathbf{Y} \in [0,1]^{L\times M}$. However, now we will also want to enforce the constraint that the rows of $\mathbf{Y}$ lie in the probabilistic simplex, while the columns should obey cardinality constraints. Denote the space of matrices for which these constraints hold as $\tilde{\mathcal{Z}}$.

Our approach translates into projection onto the intersection of two sets $\mathcal{C}$ and $\mathcal{D}$, such that $\tilde{\mathcal{Z}} = \mathcal{C} \cap \mathcal{D}$, where,

$$\mathcal{C} = \{\mathbf{Y} | \forall i,j. Y_{i,j} \geq 0, \forall i. \sum_j Y_{i,j} = 1\}$$

$$\mathcal{D} = \{\mathbf{Y} | \forall i,j. Y_{i,j} \geq 0, \forall j. \sum_i Y_{i,j} = z_j\}$$

where z_j is the cardinality constraint of class j. Since $\mathcal{C}$ and $\mathcal{D}$ are convex, we can again apply Dykstra's algorithm, alternating between the projections onto the row constraints and column constraints.

Since there is no dependence between the rows within $\mathcal{C}$, or columns within $\mathcal{D}$, the projections can be applied in parallel. Specifically, we can alternately apply simultaneous projection of all rows onto the unit simplex, and of all columns onto the positive simplex (using Algorithm 1 with $z = 1$ for each row, z_j for each column j). We note that it is also possible to discard the inequality constraint in $\mathcal{D}$ to obtain a simpler projection algorithm.

For cardinality prediction, we could learn a predictor $h_j(\mathbf{x})$ for each class j, so that we can then apply the projection operator of $\mathcal{D}$ for $z_j = h_j(\mathbf{x})$. Thus, we utilize the benefits of first predicting the desired cardinalities of each class for a given input and then projecting the columns of $\mathbf{Y}$ onto the cardinality constrained space, which have have shown successful in our experiments for the binary label case.

This extension of our framework generalizes our approach to be useful in a wide range of application domains. However, we did not experiment with this extension, and it remains an interesting direction for future work.

8. Conclusion

This paper presents a method for using highly non-linear score functions for structured prediction augmented with cardinality constraints. We show how this can be done in an end-to-end manner, by using the algorithmic form of projection into the linear constraints that restrict cardinality. This results in a powerful new structured prediction model, that can capture elaborate dependencies between the labels, and can be trained to optimize model accuracy directly.

We evaluate our method on standard datasets in multi-label classification. Our experiments demonstrate that our method achieves new state of the art results on these datasets, and outperforms all recent deep learning approaches to the problem.

We introduced the novel concept of Predict and Constrain, which we hope to be further explored in the future and applied to additional application domains. The general underlying approach is to consider high order constraints where the value of the constraint is predicted by one network, and another network implements projecting on this constraint. It will be interesting to explore other types of constraints and projections where this is possible from an algorithmic perspective and effective empirically.

Acknowledgements

This work was supported by the ISF Centers of Excellence grant, and by the Yandex Initiative in Machine Learning.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Amos, B. and Kolter, J. Z. Input-convex deep networks. 2017.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., and de Freitas, N. Learning to learn by gradient descent by gradient descent. 2016.

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

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

Boyle, J. P. and Dykstra, R. L. A method for finding projections onto the intersection of convex sets in hilbert spaces. In *Advances in order restricted statistical inference*. Springer, 1986.

Chen, L.-C., Schwing, A., Yuille, A., and Urtasun, R. Learning deep structured models. In *ICML*, 2015.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. Efficient projections onto the l 1-ball for learning in high dimensions. In *ICML*, 2008.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Escalante, R. and Raydan, M. *Alternating Projection Methods*. SIAM, 2011.

Ganchev, K., Gillenwater, J., Taskar, B., et al. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11(Jul):2001–2049, 2010.

Greff, K., Srivastava, R. K., and Schmidhuber, J. Highway and residual networks learn unrolled iterative estimation. *ICLR*, 2017.

Gupta, M. D., Kumar, S., and Xiao, J. L1 projections with box constraints. *CoRR, abs:1010.0141v1*, 2010.

Gupta, R., Diwan, A. A., and Sarawagi, S. Efficient inference with cardinality-based clique potentials. In *Proceedings of the 24th international conference on Machine learning*, pp. 329–336. ACM, 2007.

Gygli, M., Norouzi, M., and Angelova, A. Deep value networks learn to evaluate and iteratively refine structured outputs. 2017.

Hajimirsadeghi, H., Yan, W., Vahdat, A., and Mori, G. Visual recognition by counting instances: A multi-instance cardinality potential kernel. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2596–2605, 2015.

Larsson, M., Arnab, A., Kahl, F., Zheng, S., and Torr, P. H. A projected gradient descent method for crf inference allowing end-to-end training of arbitrary pairwise potentials. In *Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, 2017.

Ma, X. and Hovy, E. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL*, 2016.

Maclaurin, D., Duvenaud, D., and Adams, R. Gradient-based hyperparameter optimization through reversible learning. In *ICML*, 2015.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. Unrolled generative adversarial networks. *ICLR*, 2016.

Milch, B., Zettlemoyer, L. S., Kersting, K., Haimes, M., and Kaelbling, L. P. Lifted probabilistic inference with counting formulas. *AAAI*, pp. 10621068, 2008.

Schwing, A. G. and Urtasun, R. Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*, 2015.

Swersky, K., Sutskever, I., Tarlow, D., Zemel, R. S., Salakhutdinov, R. R., and Adams, R. P. Cardinality restricted boltzmann machines. In *NIPS*, pp. 3293–3301, 2012.

Tarlow, D., Givoni, I. E., and Zemel, R. S. Hop-map: Efficient message passing with high order potentials. In *Proceedings of 13th Conference on Artificial Intelligence and Statistics*, 2010.

Tarlow, D., Swersky, K., Zemel, R. S., Adams, R. P., and Frey, B. J. Fast exact inference for recursive cardinality models. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, 2012.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. Conditional random fields as recurrent neural networks. 2015.