Proximal Deep Structured Models

Shenlong Wang, Sanja Fidler, & Raquel Urtasun

University of Toronto

NIPS 2016
Presenter: Jack Lanchantin

Outline

Structured Prediction

- Many problems in real-world applications involve predicting a collection of random variables that are statistically related
- Graphical models have been widely exploited to encode these interactions, but they are shallow and only a log linear combination of hand-crafted– features is learned

Structured Prediction

- Deep structured models attempt to learn complex features by taking into account the dependencies between the output variables. A variety of methods have been developed in the context of predicting discrete outputs
- However, little to no attention has been given to deep structured models with continuous valued output variables.
    - One of the main reasons is that inference is much less well studied, and very few solutions exist

Continuous-Valued Structured Prediction

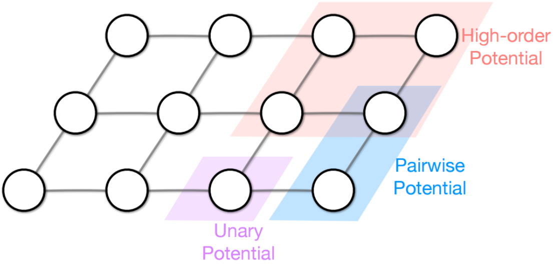
- Given input $x \in \mathcal{X}$, let $y = (y_1, ..., y_n)$ be the set of random variables we want to predict. The output space is a product space of all the elements $y \in \mathcal{Y} = \prod_{i=1}^{N} \mathcal{Y}_i$, $\mathcal{Y}_i \subset \mathbb{R}$
- $E(x, y; w)$ is an energy function which encodes the problem:

$$E(x, y; w) = \sum_i f_i(y_i, x; w_u) + \sum_\alpha f_\alpha(y_\alpha, x, w_\alpha) \tag{1}$$

where $f_i(y_i : x, w_u) : \mathcal{Y}_i \times \mathcal{X} \rightarrow \mathbb{R}$ is a function that depends on a single variable (i.e. unary term) and $f_\alpha(y_i) : \mathcal{Y}_\alpha \times \mathcal{X} \rightarrow \mathbb{R}$ depends on a subset of variables $y_\alpha = (y_i)_{i \in \alpha}$ defined on a domain $\mathcal{Y}_\alpha \subset \mathcal{Y}$

Continuous-Valued Structured Prediction

$$E(\mathbf{y}) = \underbrace{-\sum_i f_i(y_i)}_{unaries} \underbrace{- \sum_{i,j \in \mathcal{E}} f(y_i, y_j)}_{pairwise} \underbrace{- \sum_\alpha f_\alpha(\mathbf{y}_\alpha)}_{high-order}$$



Continuous-Valued Structured Prediction

- Given an input x, inference aims at finding the best configuration by minimizing the energy function:

$$y^* = argmin_{y \in \mathcal{Y}} \sum_i f_\alpha(y_i; x, w_u) + \sum_\alpha f_\alpha(y_\alpha, x, w_\alpha) \quad (2)$$

- Finding the best scoring configuration y^* is equivalent to maximizing the posteriori distribution:

$$p(y|x; w) = \frac{1}{Z(x; w)} exp(-E(x, y|w)) \quad (3)$$

Inference in Deep Structured Prediction

- Performing inference in MRFs with continuous variables involves solving a challenging numerical optimization problem
- If certain conditions are satisfied, inference is often tackled by a group of algorithms called proximal methods
- In this paper, they use proximal methods and show that it results in a particular type of recurrent net

Outline

Proximal Methods

The proximal operator $prox_f(x_0)$: $\mathbb{R} \to \mathbb{R}$ of a function is defined as:

$$prox_f(x_0) = argmin_y (y - x_0)^2 + f(y) \tag{4}$$

This involves solving a convex optimization problem, but usually there is a closed-form solution.

Outline

In order to apply proximal algorithms to tackle the inference problem defined in Eq. (2), we require the energy functions f_i and f_α to satisfy the following conditions:

1. There exist functions h_i and g_i s.t. $f_i(y_i, x; w) = g_i(y_i, h_i(x, w))$, where g_i is a distance function
2. There exists a closed-form proximal operator for $g_i(y_i, h_i(x, w))$ wrt y_i
3. There exist functions h_α and g_α s.t. $f_\alpha(y_\alpha, x; w)$ can be re-written as $f_\alpha(y_\alpha, x; w) = h_\alpha(x; w) g_\alpha(w_\alpha^T y_\alpha)$
4. There exists a proximal operator for $g_\alpha()$

Proximal Deep Structured Models

If our potential functions satisfy the conditions above, we can rewrite our objective function as follows

$$E(x, y; w) = \sum_i g_i(y_i, h_i(x; w)) + \sum_\alpha h_\alpha(x; w) g_\alpha(w_\alpha^T y_\alpha) \qquad (5)$$

Primal Dual Solvers

The general idea of primal dual solvers is to introduce auxiliary variables z to decompose the high order terms. We can then minimize z and y alternately through computing their proximal operator:

$$
\begin{aligned}
&min_{y\,\in\mathcal{Y}} max_{z\,\in\mathcal{Z}} \sum_i g_i(y_i, h_i(x; w)) \\
&- \sum_\alpha h_\alpha(x, w) g^*_\alpha(w_\alpha^T y_\alpha) + \sum_\alpha h_\alpha(x, w)\langle w_\alpha^T y_\alpha, z_\alpha\rangle
\end{aligned} \tag{6}
$$

where g^*_α is the convex conjugate of g^*

Outline

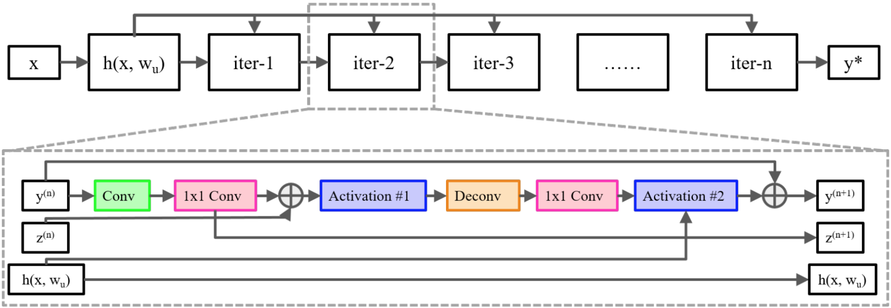
Solving Deep Structured Models

The primal-dual method solves the problem in Eq.(6) by iterating the following steps: (i) fix y and minimize the energy wrt z; (ii) fix z and minimize the energy wrt y; (iii) conduct a Nesterov extrapolation gradient step:

$$\begin{cases} z_\alpha^{(t+1)} & = & \boxed{\mathrm{prox}_{g_\alpha^*}}\left(z_\alpha^{(t)} + \boxed{\frac{\sigma_\rho}{h_\alpha(\mathbf{x};\mathbf{w})}}\boxed{\mathbf{w}_\alpha^T \bar{\mathbf{y}}_\alpha^{(t)}}\right) \\ y_i^{(t+1)} & = & \boxed{\mathrm{prox}_{g_i, h_i(\mathbf{x},\mathbf{w})}}\left(y_i^{(t)} - \boxed{\frac{\sigma_\tau}{h_\alpha(\mathbf{x};\mathbf{w})}}\boxed{\mathbf{w}_{\cdot,i}^{*T}\mathbf{z}^{(t+1)}}\right) \\ \bar{y}_i^{(t+1)} & = & y_i^{(t+1)} + \boxed{\sigma_{ex}}\left(y_i^{(t+1)} - y_i^{(t)}\right) \end{cases}$$

where $y^{(t)}$ is the solution at the t-th iteration, $z^{(t)}$ is an auxiliary variable and $h(x, w_u)$ is the deep unary network

Solving Deep Structured Models



$$
\begin{cases}
z_\alpha^{(t+1)} = \boxed{\mathrm{prox}_{g^*_\alpha}}\left(z_\alpha^{(t)} + \boxed{\frac{\sigma_\rho}{h_\alpha(\mathbf{x};\mathbf{w})}}\boxed{\mathbf{w}_\alpha^T \bar{\mathbf{y}}_\alpha^{(t)}}\right) \\
y_i^{(t+1)} = \boxed{\mathrm{prox}_{g_i, h_i(\mathbf{x},\mathbf{w})}}\left(y_i^{(t)} - \boxed{\frac{\sigma_\tau}{h_\alpha(\mathbf{x};\mathbf{w})}}\boxed{\mathbf{w}_{\cdot,i}^{*T}\mathbf{z}^{(t+1)}}\right) \\
\bar{y}_i^{(t+1)} = y_i^{(t+1)} + \boxed{\sigma_{ex}}\left(y_i^{(t+1)} - y_i^{(t)}\right)
\end{cases}
$$

Learning

Given training pairs composed of inputs $\{x_n\}_{n=1}^N$ and their corresponding output $\{y_n^{gt}\}_{n=1}^N$, learning aims at finding parameters which minimizes a regularized loss function:

$$w^* = argmin_w \sum_n \ell(y_n^*, y_n^{gt}) + \gamma ||w||_2 \quad (7)$$

Where $\ell()$ is the loss, y^* is the minimizer of RNN, and γ is a scalar.

Algorithm for Learning

Algorithm: Learning Continuous-Valued Deep Structured Models
Repeat until stopping criteria

1. Forward pass to compute $h_i(\mathbf{x}, \mathbf{w})$ and $h_\alpha(\mathbf{x}, \mathbf{w})$
2. Compute $\mathbf{y}^*$ i via forward pass in Eq. (5)
3. Compute the gradient via backward pass
4. Parameter update

Figure 2: Algorithm for learning proximal deep structured models.

Outline

Image Denoising

Corrupt each image with Gaussian noise and use the following energy function to denoise:

$$\mathbf{y}^* = argmin_{\mathbf{y}\in\mathcal{Y}} \sum_i ||y_i - x_i||_2^2 + \sum_\alpha \lambda ||\mathbf{w}_{ho,\alpha}^T \mathbf{y}_\alpha||_1 \tag{8}$$

where $prox_{\ell 2}(y,\lambda) = \frac{x+\lambda y}{1+\lambda}$ and $prox_\rho^*(z) = min(|z|,1)\cdot sign(z)$

Image Denoising

	BM3D [6]	EPLL [40]	LSSC [22]	CSF [30]	RTF [29]	Ours	Ours GPU
PSNR	28.56	28.68	28.70	28.72	28.75	**28.79**	**28.79**
Time (second)	2.57	108.72	516.48	5.10	69.25	**0.23**	**0.011**

Table 1: Natural Image Denoising on BSDS dataset [23] with noise variance $\sigma = 25$.

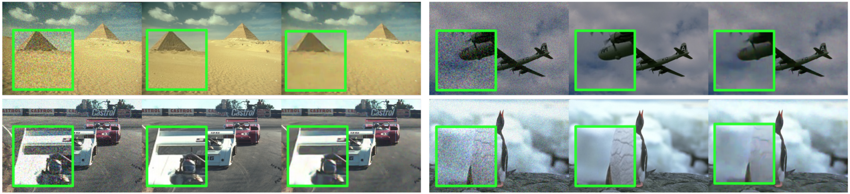


Figure 3: Qualitative results for image denoising. Left to right: noisy input, ground-truth, our result.

Depth Refinement

	Wiener	Bilateral	LMS	BM3D [6]	FilterForest [10]	Ours
PSNR	32.29	30.95	24.37	35.46	35.63	**36.31**

Table 3: Performance of depth refinement on dataset [10]

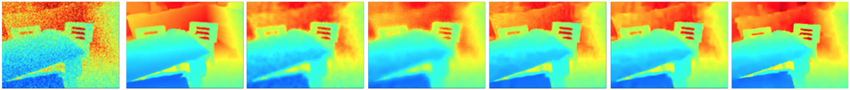


Figure 4: Qualitative results for depth refinement. Left to right: input, ground-truth, wiener filter, bilateral filter, BM3D, Filter Forest, Ours.

Outline

1. Intro
2. Deep Structured Prediction
3. Proximal Methods
   - Proximal Methods
   - Proximal Deep Structured Models
   - Solving Proximal Deep Structured Models
4. Experiments and Results
   - Image Denoising/Depth Refinement
   - Optical Flow

Optical Flow

Predict the motion between two image frames for each pixel

$$\mathbf{y}^* = argmin_{\mathbf{y} \in \mathcal{Y}} \sum_i ||y_i - f_i(x^l, x^r, w_u)||_1 + \sum_\alpha \lambda ||\mathbf{w}_{ho,\alpha}^T \mathbf{y}_\alpha||_1 \tag{9}$$

Optical Flow

	Flownet	Flownet + TV-l1	Our proposed
End-point-error	4.98	4.96	**4.91**

Table 4: Performance of optical flow on Flying chairs dataset [11]

Figure 5: **Optical flow:** Left to right: first and second input, ground-truth, Flownet [11], ours.