Learning Deep Structured Models

L.-C. Chen*, A. G. Schwing*, A. L. Yuille, R. Urtasun

* equal contribution

ICML 2015
Scene understanding

\[ x = \text{image} \]

\[ s \in S : \text{room layout} \]
Scene understanding

\[ x = \text{image} \]

\[ s \in S : \text{room layout} \]

Tag prediction

\[ x = \text{image} \]

\[ s \in S : \text{tag combos} \]
Scene understanding

\[ x = \text{image} \]

\[ s \in S : \text{room layout} \]

Tag prediction

\[ x = \text{image} \]

\[ s \in S : \text{tag combos} \]

Segmentation

\[ x = \text{image} \]

\[ s \in S : \text{segmentation} \]
Scene understanding

$\mathbf{x} = \text{image}$

$s \in S : \text{room layout}$

Tag prediction

$\mathbf{x} = \text{image}$

$s \in S : \text{tag combos}$

Segmentation

$\mathbf{x} = \text{image}$

$s \in S : \text{segmentation}$

\[ s^* = \arg \max_{s \in S} F(s, x, w) \]
How to find the parameters $w$ of the scoring function $F(s, x, w)$?
Learning

good parameters from annotated examples

\[ D = \{(x, s)\}\]

- Log-linear models (CRFs, structured SVMs):
  \[ F(s, x, w) = w^\top \tilde{F}(s, x) \]

- Non-linear models (CNNs):
  \[ F(s, x, w) \]
Inference:

\[ s^* = \arg \max_{s \in S} F(s, x, w) \]
Inference:

\[ s^* = \arg \max_{s \in S} F(s, x, w) \]

Probability of a configuration \( s \):

\[ p(s \mid x, w) = \frac{1}{Z(x, w)} \exp F(s, x, w) \]

\[ Z(x, w) = \sum_{\hat{s} \in S} \exp F(\hat{s}, x, w) \]
Inference:

\[ s^* = \arg \max_{s \in S} F(s, x, w) \]

Probability of a configuration \( s \):

\[ p(s \mid x, w) = \frac{1}{Z(x, w)} \exp F(s, x, w) \]

\[ Z(x, w) = \sum_{\hat{s} \in S} \exp F(\hat{s}, x, w) \]

Equivalently:

\[ s^* = \arg \max_{s \in S} p(s \mid x, w) \]
Probability of a configuration $s$:

$$p(s \mid x, w) = \frac{1}{Z(x, w)} \exp F(s, x, w)$$

$$Z(x, w) = \sum_{\hat{s} \in S} \exp F(\hat{s}, x, w)$$
Probability of a configuration $s$:

$$p(s \mid x, w) = \frac{1}{Z(x, w)} \exp F(s, x, w)$$

$$Z(x, w) = \sum_{\hat{s} \in S} \exp F(\hat{s}, x, w)$$

Maximize the likelihood of training data via

$$w^* = \arg \max_w \prod_{(x, s) \in D} p(s \mid x, w)$$
Probability of a configuration $s$:

$$p(s \mid x, w) = \frac{1}{Z(x, w)} \exp F(s, x, w)$$

$$Z(x, w) = \sum_{\hat{s} \in S} \exp F(\hat{s}, x, w)$$

Maximize the likelihood of training data via

$$w^* = \arg \max_w \prod_{(x,s) \in \mathcal{D}} p(s| x, w)$$

$$= \arg \max_w \sum_{(x,s) \in \mathcal{D}} \left( F(s, x, w) - \ln \sum_{\hat{s} \in S} \exp F(\hat{s}, x, w) \right)$$
\[
\max_w \sum_{(x, s) \in \mathcal{D}} \left( F(s, x, w) - \ln \sum_{\hat{s} \in \mathcal{S}} \exp F(\hat{s}, x, w) \right)
\]
\[
\max_w \sum_{(x,s) \in \mathcal{D}} \left( F(s, x, w) - \ln \sum_{\hat{s} \in \mathcal{S}} \exp F(\hat{s}, x, w) \right)
\]

Optimize via gradient ascent

\[
\frac{\partial}{\partial w} \left( F(s, x, w) - \ln \sum_{\hat{s} \in \mathcal{S}} \exp F(\hat{s}, x, w) \right)
\]
\[
\max_w \sum_{(x,s) \in D} \left( F(s, x, w) - \ln \sum_{\hat{s} \in S} \exp F(\hat{s}, x, w) \right)
\]

Optimize via gradient ascent

\[
\frac{\partial}{\partial w} \left( F(s, x, w) - \ln \sum_{\hat{s} \in S} \exp F(\hat{s}, x, w) \right)
\]

\[
= \sum_{(x,s) \in D, \hat{s}} \left( \delta(\hat{s} = s) - p(\hat{s} | x; w) \right) \frac{\partial}{\partial w} F(\hat{s}, x, w)
\]
\[
\max_w \sum_{(x,s) \in \mathcal{D}} \left( F(s, x, w) - \ln \sum_{\hat{s} \in \mathcal{S}} \exp F(\hat{s}, x, w) \right)
\]

Optimize via gradient ascent

\[
\frac{\partial}{\partial w} \left( F(s, x, w) - \ln \sum_{\hat{s} \in \mathcal{S}} \exp F(\hat{s}, x, w) \right) = \sum_{(x,s) \in \mathcal{D}, \hat{s}} \left( \delta(\hat{s} = s) - p(\hat{s} \mid x; w) \right) \frac{\partial}{\partial w} F(\hat{s}, x, w)
\]

- Compute predicted distribution \( p(\hat{s} \mid x; w) \)
- Use chain rule to pass back difference between prediction and observation
Algorithm: Deep Learning

Repeat until stopping criteria

1. Forward pass to compute $F(s, x, w)$
2. Compute $p(s | x, w)$ via soft-max
3. Backward pass via chain rule to obtain gradient
4. Update parameters $w$

Where are the challenges?
Algorithm: Deep Learning

Repeat until stopping criteria

1. Forward pass to compute $F(s, x, w)$
2. Compute $p(s \mid x, w)$ via soft-max
3. Backward pass via chain rule to obtain gradient
4. Update parameters $w$

Where are the challenges?

- How do we even represent $F(s, x, w)$ if $S$ is large?
- How do we compute $p(s \mid x, w)$?
Algorithm: Deep Learning

Repeat until stopping criteria

1. Forward pass to compute $F(s, x, w)$
2. Compute $p(s \mid x, w)$ via soft-max
3. Backward pass via chain rule to obtain gradient
4. Update parameters $w$

Where are the challenges?

- How do we even represent $F(s, x, w)$ if $S$ is large?
- How do we compute $p(s \mid x, w)$?

A solution to deal with large output spaces

(Citation: Chen, Schwing, Yuille, Urtasun () Learning Deep Structured Models 2015 8 / 22)
Output space size of motivating applications:

Scene understanding

\[ |S| = 50^4 \]
Output space size of motivating applications:

Scene understanding  Tag prediction

\[ |S| = 50^4 \quad \text{and} \quad |S| = 2^{\# \text{tags}} \]
Output space size of motivating applications:

Scene understanding: $|S| = 50^4$

Tag prediction: $|S| = 2^{\text{#tags}}$

Segmentation: $|S| = C^{\text{#pixels}}$
Output space size of motivating applications:

- Scene understanding
- Tag prediction
- Segmentation

\[ |S| = 50^4 \quad |S| = 2^{\#\text{tags}} \quad |S| = C^{\#\text{pixels}} \]

Observation:
- Interest in jointly predicting multiple variables \( s = (s_1, \ldots, s_n) \)
Output space size of motivating applications:

Scene understanding  Tag prediction  Segmentation

|S| = 50^4  |S| = 2^\#tags  |S| = C^\#pixels

Observation:
- Interest in jointly predicting multiple variables \( s = (s_1, \ldots, s_n) \)

Solution:
- Assume scoring function to decomposes additively

\[
F(s, x, w) = F(s_1, \ldots, s_n, x, w) = \sum_r f_r(s_r, x, w)
\]
Output space size of motivating applications:

Scene understanding  Tag prediction  Segmentation

\[ |S| = 50^4 \quad |S| = 2^{\#tags} \quad |S| = C^{\#pixels} \]

Observation:
- Interest in jointly predicting multiple variables \( s = (s_1, \ldots, s_n) \)

Solution:
- Assume scoring function to decomposes additively

\[
F(s, x, w) = F(s_1, \ldots, s_n, x, w) = \sum_r f_r(s_r, x, w)
\]

Every \( f_r(s_r, x, w) \) is an arbitrary composite function, e.g., a CNN
\[
F(s, x, w) = F(s_1, \ldots, s_n, x, w) = \sum_r f_r(s_r, x, w)
\]
$$F(s, x, w) = F(s_1, \ldots, s_n, x, w) = \sum_r f_r(s_r, x, w)$$

How to compute gradient:

$$\frac{\partial}{\partial w} \left( F(s, x, w) - \ln \sum_{\hat{s} \in S} \exp F(\hat{s}, x, w) \right)$$

$$= \sum_{(x,s) \in D, \hat{s}} (\delta(\hat{s} = s) - p(\hat{s} | x; w)) \frac{\partial}{\partial w} F(\hat{s}, x, w)$$

How to obtain marginals $p_r(\hat{s} | x, w)$?

Approximate marginals $b_r(\hat{s} | x, w)$ via:

Sampling techniques

Variational methods (e.g., blending techniques)
\[ F(s, x, w) = F(s_1, \ldots, s_n, x, w) = \sum_r f_r(s_r, x, w) \]

How to compute gradient:

\[
\frac{\partial}{\partial w} \left( F(s, x, w) - \ln \sum_{\hat{s} \in S} \exp F(\hat{s}, x, w) \right) \\
= \sum_{(x,s) \in D, \hat{s}} \left( \delta(\hat{s} = s) - p(\hat{s} | x; w) \right) \frac{\partial}{\partial w} F(\hat{s}, x, w) \\
= \sum_{(x,s) \in D, r, \hat{s}_r} \left( \delta_r(\hat{s}_r = s_r) - p_r(\hat{s}_r | x; w) \right) \frac{\partial}{\partial w} f_r(\hat{s}_r, x, w)
\]
\[ F(s, x, w) = F(s_1, \ldots, s_n, x, w) = \sum_r f_r(s_r, x, w) \]

How to compute gradient:

\[
\frac{\partial}{\partial w} \left( F(s, x, w) - \ln \sum_{\hat{s} \in S} \exp F(\hat{s}, x, w) \right) = \sum_{(x,s) \in D, \hat{s}} \left( \delta(\hat{s} = s) - p(\hat{s} | x; w) \right) \frac{\partial}{\partial w} F(\hat{s}, x, w)
\]

\[
= \sum_{(x,s) \in D, \hat{s}} \left( \delta_r(\hat{s}_r = s_r) - p_r(\hat{s}_r | x; w) \right) \frac{\partial}{\partial w} f_r(\hat{s}_r, x, w)
\]

How to obtain marginals \( p_r(\hat{s}_r | x, w) \)?
\[ F(s, x, w) = F(s_1, \ldots, s_n, x, w) = \sum_r f_r(s_r, x, w) \]

How to compute gradient:

\[
\frac{\partial}{\partial w} \left( F(s, x, w) - \ln \sum_{\hat{s} \in S} \exp F(\hat{s}, x, w) \right)
\]

\[
= \sum_{(x,s) \in \mathcal{D}, \hat{s}} \left( \delta(\hat{s} = s) - p(\hat{s} | x; w) \right) \frac{\partial}{\partial w} F(\hat{s}, x, w)
\]

\[
= \sum_{(x,s) \in \mathcal{D}, r, \hat{s}_r} \left( \delta_r(\hat{s}_r = s_r) - p_r(\hat{s}_r | x; w) \right) \frac{\partial}{\partial w} f_r(\hat{s}_r, x, w)
\]

How to obtain marginals \( p_r(\hat{s}_r | x, w) \)?

Approximate marginals \( b_r(\hat{s}_r | x, w) \) via:

- Sampling techniques
- Variational methods (e.g., blending techniques)
Standard learning:

repeat
  repeat
    update marginals $b_r$
    until convergence
  update parameters $w$
  until convergence

Blended learning:

repeat
  update marginals $b_r$
  update parameters $w$
  until convergence

Advantage

More frequent parameter updates
Intuition: Standard CNN

$S_1$

CNN
Intuition: Independent Prediction

\[ S_1 \quad CNN_1 \]

\[ S_2 \quad CNN_2 \]
Intuition: Deep Structured Learning

$S_{1,2}$

$S_{1}$

$S_{2}$

$S_{2,3}$

$S_{3}$

$\text{CNN}_1$

$\text{CNN}_2$

$\text{CNN}_3$

$\text{CNN}_4$

$\text{CNN}_5$
Approximated Deep Structured Learning

[Domke'13; Deng et al.'14; Tompson et al.'14; Li&Zemel'14; Chen et al.'15]

Sample parallel implementation:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Each compute node uses GPU for CNN Forward pass to compute $f_r(\hat{s}_r, x, w)$</td>
</tr>
<tr>
<td>2</td>
<td>Each compute node estimates beliefs $b_r(\hat{s}_r</td>
</tr>
<tr>
<td>3</td>
<td>Backpropagation of difference using GPU to obtain machine local gradient</td>
</tr>
<tr>
<td>4</td>
<td>Synchronize gradient across all machines using MPI</td>
</tr>
<tr>
<td>5</td>
<td>Update parameters $w$</td>
</tr>
</tbody>
</table>
Dealing with large number $|\mathcal{D}|$ of training examples:

- Usage of mini batches
- Parallelized across samples (any number of machines and GPUs)

Dealing with large output spaces $\mathcal{S}$:

- Variational approximations
- Blending of learning and inference
Experimental Results

Two tasks:
- Predicting words from noisy images
- Tagging Flickr photographs
Word dataset

Find five letters within distorted images

- Graphical model: 1st or 2nd order Markov

- Unary function: multi-layer perceptron (MLP)
- Pairwise function: linear or non-linear MLP

\[ |S| = 26^5 \]
**Deeper and more structured ⇒ better performance**
Non-linear pairwise function improves over the linear one
Given an image assign a combination of tags that describe the content

- Graphical model: $K_{38}$
- Unary function: AlexNet
- Pairwise function: linear
- $|S| = 2^{38}$
- 10000 training examples
- 10000 test examples

<table>
<thead>
<tr>
<th>Training method</th>
<th>Prediction error [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unary only</td>
<td>9.36</td>
</tr>
<tr>
<td>Piecewise</td>
<td>7.70</td>
</tr>
<tr>
<td>Joint (with pre-training)</td>
<td>7.25</td>
</tr>
</tbody>
</table>
female/indoor/portrait
female/indoor/portrait
skys/plant life/tree
sky/plant life/tree
water/animals/sea
water/animals/sky
animals/dog/indoor
animals/dog
indoor/flower/plant life
∅
Learned class correlations:

<table>
<thead>
<tr>
<th></th>
<th>female</th>
<th>people</th>
<th>indoor</th>
<th>portrait</th>
<th>sky</th>
<th>plant life</th>
<th>male</th>
<th>clouds</th>
<th>tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>female</td>
<td>0.00</td>
<td>0.68</td>
<td>0.04</td>
<td>0.24</td>
<td>-0.01</td>
<td>-0.05</td>
<td>0.07</td>
<td>-0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>people</td>
<td>0.68</td>
<td>0.00</td>
<td>0.06</td>
<td>0.36</td>
<td>-0.05</td>
<td>-0.12</td>
<td>0.74</td>
<td>-0.04</td>
<td>-0.03</td>
</tr>
<tr>
<td>indoor</td>
<td>0.04</td>
<td>0.06</td>
<td>0.00</td>
<td>0.07</td>
<td>-0.35</td>
<td>-0.34</td>
<td>0.02</td>
<td>-0.15</td>
<td>-0.21</td>
</tr>
<tr>
<td>portrait</td>
<td>0.24</td>
<td>0.36</td>
<td>0.07</td>
<td>0.00</td>
<td>-0.02</td>
<td>-0.01</td>
<td>0.12</td>
<td>0.02</td>
<td>0.05</td>
</tr>
<tr>
<td>sky</td>
<td>-0.01</td>
<td>-0.05</td>
<td>-0.35</td>
<td>-0.02</td>
<td>0.00</td>
<td>0.24</td>
<td>-0.00</td>
<td>0.44</td>
<td>0.30</td>
</tr>
<tr>
<td>plant life</td>
<td>-0.05</td>
<td>-0.12</td>
<td>-0.34</td>
<td>-0.01</td>
<td>0.24</td>
<td>0.00</td>
<td>-0.07</td>
<td>0.09</td>
<td>0.68</td>
</tr>
<tr>
<td>male</td>
<td>0.07</td>
<td>0.74</td>
<td>0.02</td>
<td>0.12</td>
<td>-0.00</td>
<td>-0.07</td>
<td>0.00</td>
<td>0.00</td>
<td>-0.02</td>
</tr>
<tr>
<td>clouds</td>
<td>-0.01</td>
<td>-0.04</td>
<td>-0.15</td>
<td>0.02</td>
<td>0.44</td>
<td>0.09</td>
<td>0.00</td>
<td>0.00</td>
<td>0.11</td>
</tr>
<tr>
<td>tree</td>
<td>0.01</td>
<td>-0.03</td>
<td>-0.21</td>
<td>0.05</td>
<td>0.30</td>
<td>0.68</td>
<td>-0.02</td>
<td>0.11</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Deep Learning Meets Structured Prediction

Non-linear Structured Prediction:
- Modeling of correlations between variables
- Non-linear dependence on parameters
- Joint training of many convolutional neural networks
- Parallel implementation
- Decomposition of functions
- Blended learning techniques
- Wide range of applicability

Source code available on http://alexander-schwing.de

Jay and Alex’s expertise is available on the 2016 job market!