Nonlinear Structured Prediction using the GPU: Deep Learning meets Structured Prediction

Alexander G. Schwing

in collaboration with L.-C. Chen, A. L. Yuille and R. Urtasun

GPU Technology Conference, March 18, 2015

Classification

$x = \text{image}$

$s \in \mathcal{S}$: categories

Classification

$x =$ image

$s \in \mathcal{S}$: categories

Scene understanding

$x =$ image

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

Classification

$x = \text{image}$

$s \in \mathcal{S}$: categories

Scene understanding

$x = \text{image}$

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

Tag prediction

$x = \text{image}$

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

Classification

$x = \text{image}$

$s \in \mathcal{S}$: categories

Scene understanding

$x = \text{image}$

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

Tag prediction

$x = \text{image}$

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

Segmentation

$x = \text{image}$

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

Inference

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

Challenge: The domain size $|\mathcal{S}|$ is potentially large

- ImageNet classification: $|\mathcal{S}| = 1000$
- Scene understanding: $|\mathcal{S}| = 50^4$
- Tag prediction: $|\mathcal{S}| = 2^{\text{Number of tags}}$
- Image segmentation: $|\mathcal{S}| = C^{\text{Number of pixels}}$

Computation of $F(s, x, w)$ for all possible $s \in \mathcal{S}$ in general often intractable.

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

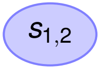
Assumption: function/model decomposes additively

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

- Restriction r: $s_r = (s_i)_{i \in r}$
- Discrete domain:

$$f_{\{1,2\}}(s_{\{1,2\}}) = f_{\{1,2\}}(s_1, s_2) = \left[f_{\{1,2\}}(1,1), f_{\{1,2\}}(1,2), \ldots\right]$$

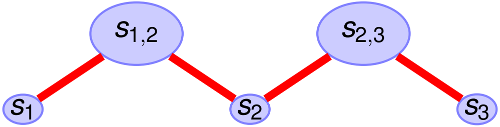
- Visualization



[Werner'07; Wainwright&Jordan'08; Koller&Friedman'09]

Example

$$s^* = \arg\max_s f_1(s_1, x, w) + f_2(s_2, x, w) + f_3(s_3, x, w) + f_{1,2}(s_{1,2}, x, w) + f_{2,3}(s_{2,3}, x, w)$$



Dual decomposition techniques for distributed inference

How to find the parameters w of the scoring function $F(s, x, w)$?

Learning

[Lafferty et al.'01; Taskar et al.'03; Joachims et al.'04; Hinton et al.'84; Bengio et al.'89 ; LeCun et al.'89; Salakhutdinov et al.'08; Lee et al.'09; Fergus et al.'10;]

good parameters from annotated examples

$$\mathcal{D} = \{(x, s)\}$$

- Log-linear models (CRFs, structured SVMs):

$$F(s, x, w) = w^\top \tilde{F}(s, x)$$

- Non-linear models, e.g., CNNs (this talk):

$$F(s, x, w)$$

Learning

[Lafferty et al.'01; Taskar et al.'03; Joachims et al.'04; Hinton et al.'84; Bengio et al.'89 ; LeCun et al.'89; Salakhutdinov et al.'08; Lee et al.'09; Fergus et al.'10;]

good parameters from annotated examples

$$\mathcal{D} = \{(x, s)\}$$

- Log-linear models (CRFs, structured SVMs):

$$F(s, x, w) = w^\top \tilde{F}(s, x)$$

- Non-linear models, e.g., CNNs (this talk):

$$F(s, x, w)$$

Dealing with multiple variables using CNNs

Inference:

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

Probability of a configuration s:

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

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

Inference alternatively:

$$s^* = \arg\max_{s \in \mathcal{S}} p(s \mid x, w)$$

Probability of a configuration s:

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

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

Maximize the likelihood of training data via

$$\begin{aligned} w^* &= \arg\max_w \prod_{(x,s) \in \mathcal{D}} p(s|x,w) \\ &= \arg\max_w \sum_{(x,s) \in \mathcal{D}} \left(F(s,x,w) - \ln \sum_{\hat{s} \in \mathcal{S}} \exp F(s,x,w) \right) \end{aligned}$$

Maximum likelihood is equivalent to maximizing cross-entropy:

- Target distribution: $p_{(x,s),\text{tg}}(\hat{s}) = \delta(\hat{s} = s)$
- Cross-Entropy:

$$
\begin{aligned}
& \max_w \sum_{(x,s)\in\mathcal{D},\hat{s}} p_{(x,s),\text{tg}}(\hat{s}) \ln p(\hat{s} \mid x; w) \\
= \quad & \max_w \sum_{(x,s)\in\mathcal{D}} \ln p(s \mid x; w) \\
= \quad & \max_w \ln \prod_{(x,s)\in\mathcal{D}} p(s \mid x; w)
\end{aligned}
$$

Program of interest:

$$\max_w \sum_{(x,s)\in\mathcal{D},\hat{s}} p_{(x,s),\text{tg}}(\hat{s}) \ln p(\hat{s} \mid x; w)$$

Optimize via gradient ascent

$$\begin{aligned}
&\frac{\partial}{\partial w} \sum_{(x,s)\in\mathcal{D},\hat{s}} p_{(x,s),\text{tg}}(\hat{s}) \ln p(\hat{s} \mid x; w) \\
&= \sum_{(x,s)\in\mathcal{D},\hat{s}} \left(p_{(x,s),\text{tg}}(\hat{s}) - p(\hat{s} \mid x; w)\right) \frac{\partial}{\partial w} F(\hat{s}, x, w) \\
&= \sum_{(x,s)\in\mathcal{D}} \left(\mathbb{E}_{p_{(x,s),\text{tg}}}\left[\frac{\partial}{\partial w} F(\hat{s}, x, w)\right] - \mathbb{E}_{p_{(x,s)}}\left[\frac{\partial}{\partial w} F(\hat{s}, x, w)\right]\right)
\end{aligned}$$

- 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 $p(s \mid x, w)$
3. Backward pass via chain rule to obtain gradient
4. Update parameters w

Why are large output spaces a challenge?

Algorithm: Deep Learning

Repeat until stopping criteria

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

Why are large output spaces a challenge?

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

Domain size of typical applications:

- ImageNet classification: $|\mathcal{S}| = 1000$
- Scene understanding: $|\mathcal{S}| = 50^4$
- Tag prediction: $|\mathcal{S}| = 2^{\text{Number of tags}}$
- Image segmentation: $|\mathcal{S}| = C^{\text{Number of pixels}}$

Solution:

- Interest in jointly predicting multiple variables $s = (s_1, \ldots, s_n)$
- Assumption: function/model 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 function, e.g., a CNN

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

How to compute gradient:

$$\frac{\partial}{\partial w} \sum_{(x,s)\in\mathcal{D},\hat{s}} p_{(x,s),\text{tg}}(\hat{s}) \ln p(\hat{s} \mid x; w)$$

$$= \sum_{(x,s)\in\mathcal{D}} \left(\mathbb{E}_{p_{(x,s),\text{tg}}} \left[\frac{\partial}{\partial w} F(\hat{s},x,w) \right] - \mathbb{E}_{p_{(x,s)}} \left[\frac{\partial}{\partial w} F(\hat{s},x,w) \right] \right)$$

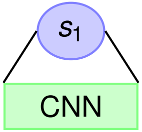
$$= \sum_{(x,s)\in\mathcal{D},r} \left(\mathbb{E}_{p_{(x,s),r,\text{tg}}} \left[\frac{\partial}{\partial w} f_r(\hat{s}_r,x,w) \right] - \mathbb{E}_{p_{(x,s),r}} \left[\frac{\partial}{\partial w} f_r(\hat{s}_r,x,w) \right] \right)$$

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

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

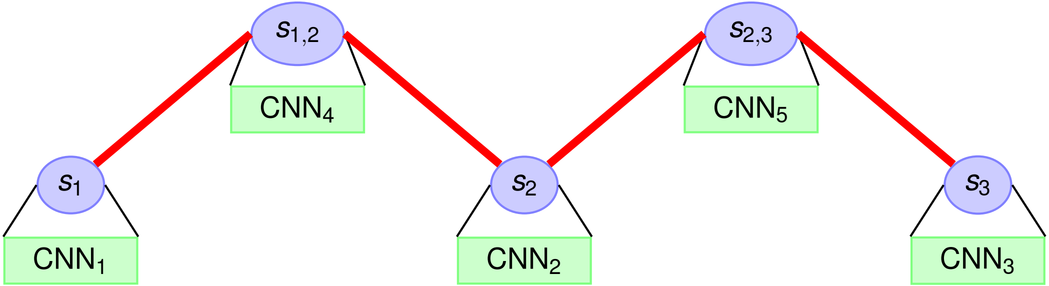
- Sampling methods
- Variational methods

Intuition: Standard CNN



Intuition: Independent Prediction

Intuition: Deep Structured Learning



Approximated Deep Structured Learning

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

Sample parallel implementation:

Partition data $\mathcal{D}$ onto compute nodes
Repeat until stopping criteria

1. Each compute node uses GPU for CNNs forward pass to compute $f_r(\hat{s}_r, x, w)$
2. Each compute node estimates beliefs $b_r(\hat{s}_r \mid x, w)$ for assigned samples
3. Backpropagation of difference using GPU to obtain machine local gradient
4. Synchronize gradient across all machines using MPI
5. Update parameters w

Dealing with large number $|\mathcal{D}|$ of training examples:

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

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

- Variational approximations
- Blending of learning and inference

ImageNet dataset

[Russakovsky et al.'14
Krizhvsky et al.'13; Simonyan&Zisserman'14; Szegedy et al.'14; Jia et al.' 14]

- Model:

- $|\mathcal{S}| = 1000$
- 1.2 million training examples
- 50,000 validation examples

Model	Top 5 validation set error [%]
AlexNet	19.95
DeepNet16	10.29
DeepNet19	10.37

Different from reported results because of missing averaging, different image crops, etc.

Visual results

Groundtruth: Alp

s_1	$F(s_1, x, w)$
Ski	85.28%
Alp	9.10%
Shovel	4.86%
King penguin	0.14%
Dogsled	0.10%

Visual results

Groundtruth: Puma

s_1	$F(s_1, x, w)$
Puma	99.87%
Lynx	0.06%
Koala	0.03%
Lion	0.01%
Egyptian cat	0.00%

Visual results

Groundtruth: Cradle

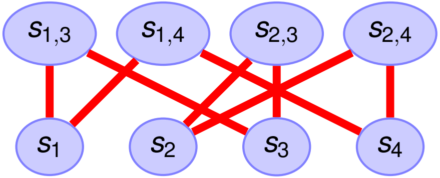
s_1	$F(s_1, x, w)$
Diaper	12.51%
Beagle	10.98%
Teddy bear	8.12%
Pajama	7.06%
Crib	5.22%

Layout dataset

[Hoiem et al.'05; Hedau et al.'09]

Given a single image x, predict a 3D parametric box that best describes the observed room layout

- $|\mathcal{S}| = 50^4$
- Linear model
- 205 training examples
- 104 test examples



Pixel-wise prediction errors [%] on layout dataset:

	OM	GC	OM + GC	Others
[Hoiem07]	-	28.9	-	-
[Hedau09]	-	21.2	-	-
[Wang10]	22.2	-	-	-
[Lee10]	24.7	22.7	18.6	-
[Pero12]	-	-	-	16.3
Ours	**18.63**	**15.35**	**13.59**	-

Flickr dataset

[Huiskes&Lew'08]

Assign a subset of 38 possible tags to a given image x

- Model: K_{38}
- $|\mathcal{S}| = 2^{38}$
- 10000 training examples
- 10000 test examples

Training method	Prediction error [%]
Unary only	9.36
Piecewise	7.70
Joint (with pre-training)	**7.25**

Visual results

female/indoor/portrait
female/indoor/portrait

sky/plant life/tree
sky/plant life/tree

water/animals/sea
water/animals/sky

indoor/flower/plant life
∅

Learned class correlations:

	female	people	indoor	portrait	sky	plant life	male	clouds	tree
female	0.00	0.68	0.04	0.24	-0.01	-0.05	0.07	-0.01	0.01
people	0.68	0.00	0.06	0.36	-0.05	-0.12	0.74	-0.04	-0.03
indoor	0.04	0.06	0.00	0.07	-0.35	-0.34	0.02	-0.15	-0.21
portrait	0.24	0.36	0.07	0.00	-0.02	-0.01	0.12	0.02	0.05
sky	-0.01	-0.05	-0.35	-0.02	0.00	0.24	-0.00	0.44	0.30
plant life	-0.05	-0.12	-0.34	-0.01	0.24	0.00	-0.07	0.09	0.68
male	0.07	0.74	0.02	0.12	-0.00	-0.07	0.00	0.00	-0.02
clouds	-0.01	-0.04	-0.15	0.02	0.44	0.09	0.00	0.00	0.11
tree	0.01	-0.03	-0.21	0.05	0.30	0.68	-0.02	0.11	0.00

Pascal VOC 2012 dataset

[Everingham et al.'14; Chen et al.'15; Long et al.'15; Zheng et al.'15
Krähenbühl&Koltun'11,'13; Vineet et al.'12]

- Model: $K_{350 \cdot 500}$
- $|\mathcal{S}| = 21^{350 \cdot 500}$
- $\approx$ 10000 training examples
- $\approx$ 1500 validation examples

Training method	Mean IoU Accuracy [%]
Unary only	61.476
Joint	**64.060**

Visual results

Thanks

- collaborators L.-C. Chen, A. L. Yuille and R. Urtasun
- NVIDIA for donating one Tesla K40 GPU

Nonlinear Structured Prediction

- Modeling of correlations between variables
- Nonlinear dependence on parameters
- Joint training of many convolutional neural networks
- Distributed onto multiple compute nodes
- Each using a GPU

http://alexander-schwing.de