

Methods for Learning Structured Prediction in Semantic Segmentation of Natural Images

Andreas C. Müller

University of Bonn

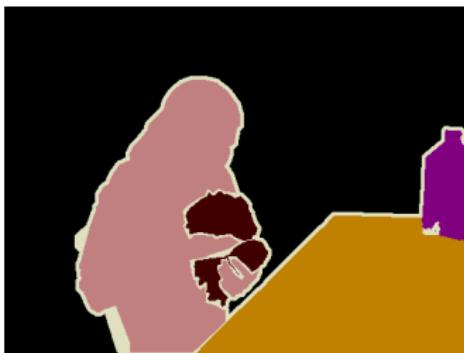
April 10, 2018

- 1 Learning Structured Prediction for Semantic Segmentation
- 2 Learning Depth-Sensitive Conditional Random Fields
- 3 Learning Loopy CRFs Exactly
- 4 PyStruct - Structured Prediction in Python

Semantic Segmentation



Semantic Segmentation



Semantic Segmentation as Structured Prediction



Semantic Segmentation as Structured Prediction



Semantic Segmentation as Structured Prediction



Semantic Segmentation as Structured Prediction



Structured Prediction

Learn prediction function of the form

$$g(x, w) := \arg \max_{y \in \mathcal{Y}} w^T \psi(x, y)$$

Structured Prediction

Learn prediction function of the form

$$g(x, w) := \arg \max_{y \in \mathcal{Y}} w^T \psi(x, y)$$

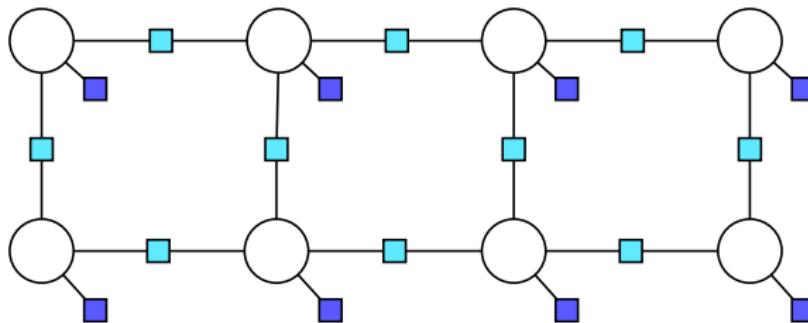
Objective

$$\min_w \frac{1}{2} \|w\|^2 + C \sum_i \ell(x^i, y^i, w)$$

$$\ell(x^i, y^i, w) = [\max_{y \in \mathcal{Y}} \Delta(y^i, y) + w^T \psi(x^i, y) - w^T \psi(x^i, y^i)]_+.$$

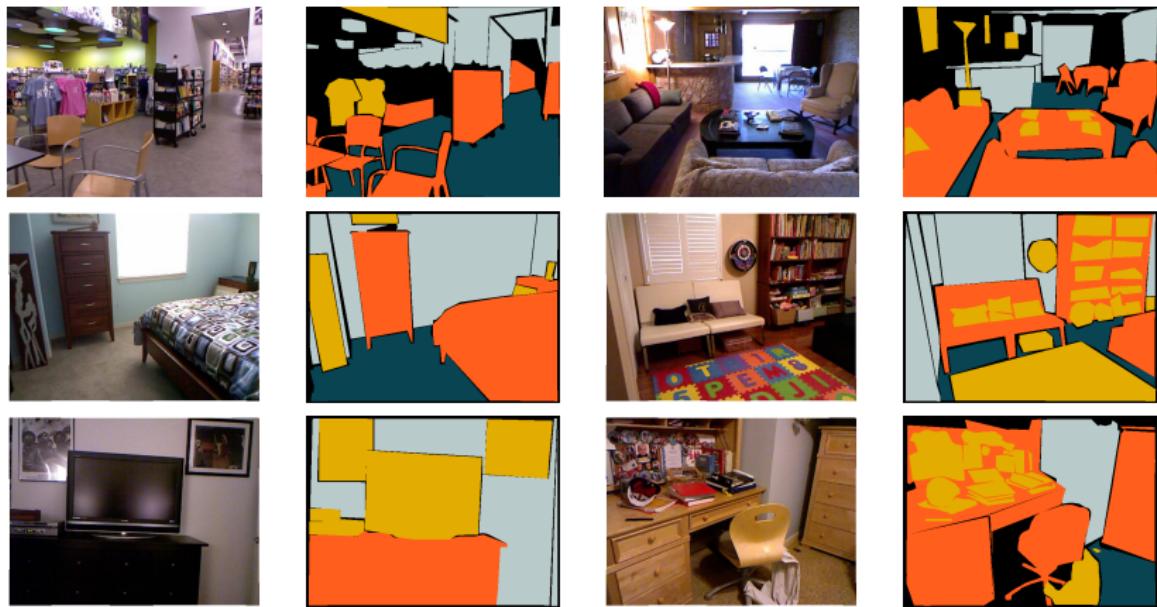
?

Pairwise Models



$$\mathbf{w}^T \psi(x, y) = \sum_{(i,j) \in E} w_{i,j} \psi_{i,j}(x, y_i, y_j) + \sum_{i \in I} w_i \psi_i(x, y_i)$$

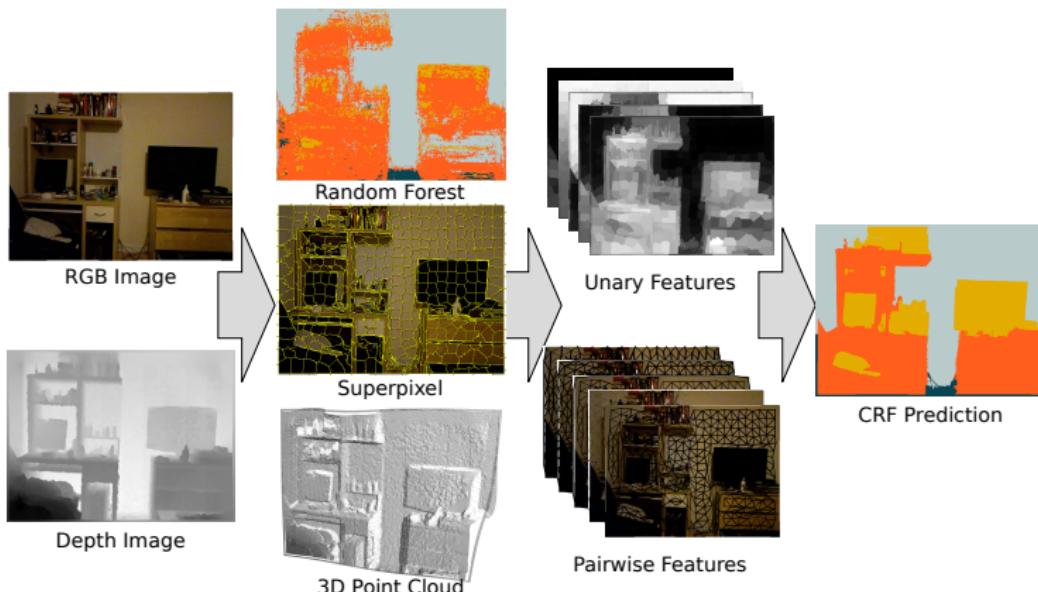
Dataset: NYUv2



■ Ground ■ Structure ■ Furniture ■ Props ■ Void

795 training images, 654 test images.

Overview

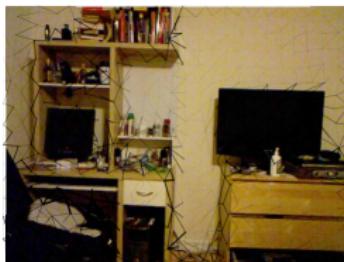


Pairwise Features

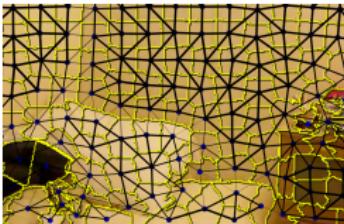
$$f_{\text{color}}(x_i, x_j) = \exp(-\gamma \|c_i - c_j\|^2)$$



$$f_{\text{depth}}(x_i, x_j) = (d_i - d_j)/Z$$



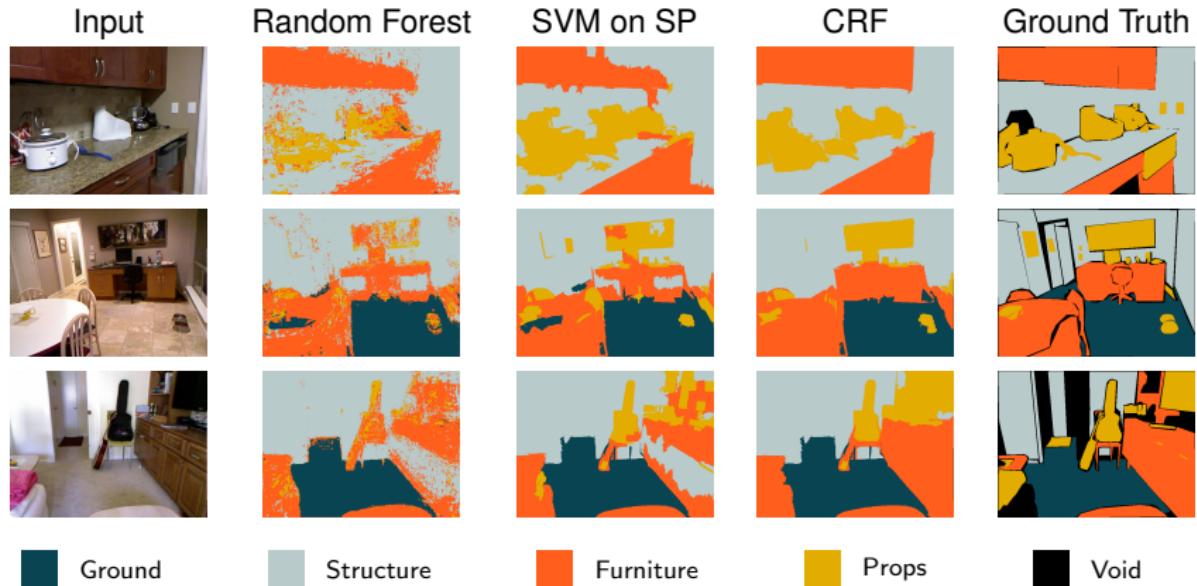
$$f_{\text{direction}}(x_i, x_j) = \langle \text{pos}_{x_i} - \text{pos}_{x_j}, [0, 1]^T \rangle \quad f_{\text{normals}}(x_i, x_j) = 1 - \frac{1}{\pi} \langle \mathbf{n}_{x_i}, \mathbf{n}_{x_j} \rangle$$



Learning and Optimization

- 1-slack SSVM
- Inference using fusion moves and AD³.
- Exact learning in loopy model (?).

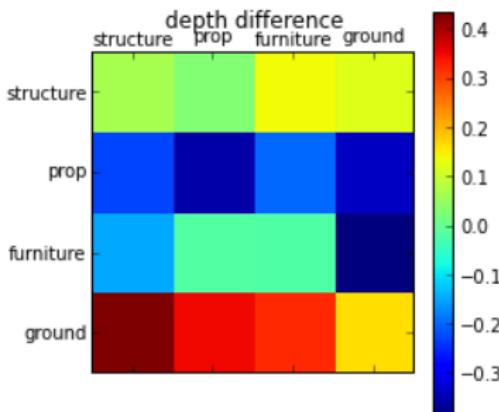
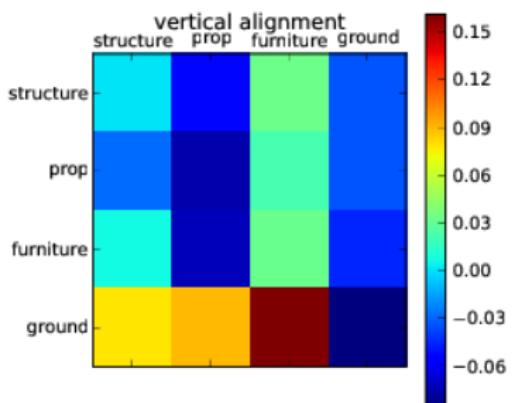
Qualitative Results



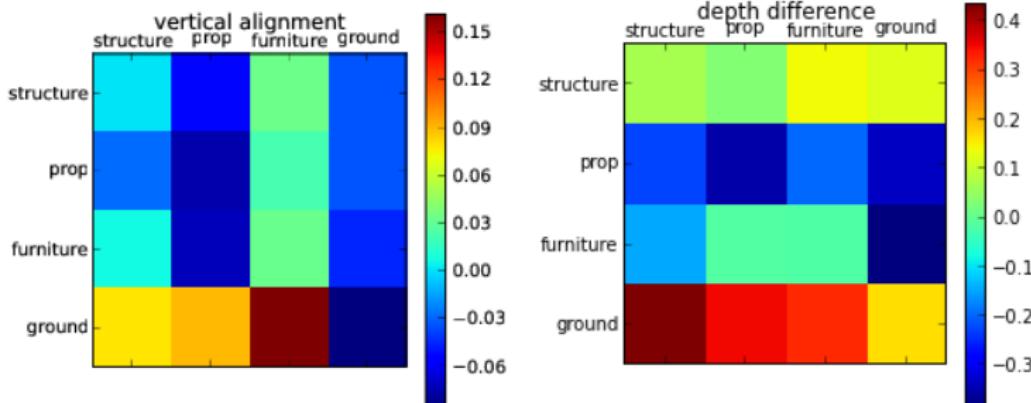
Results

	ground	structure	furniture	props	class avg	pixel avg
RF	90.8	81.6	67.9	19.9	65.0	68.3
RF + SP	92.5	83.3	73.8	13.9	65.7	70.1
RF + SP + SVM	94.4	79.1	64.2	44.0	70.4	70.3
RF + SP + CRF	94.9	78.9	71.1	42.7	71.9	72.3
?	68	59	70	42	59.6	58.6
?	87.3	86.1	45.3	35.5	63.5	64.5
? [†]	95.6	83.0	75.1	14.2	67.0	70.9

Learned Potentials



Learned Potentials

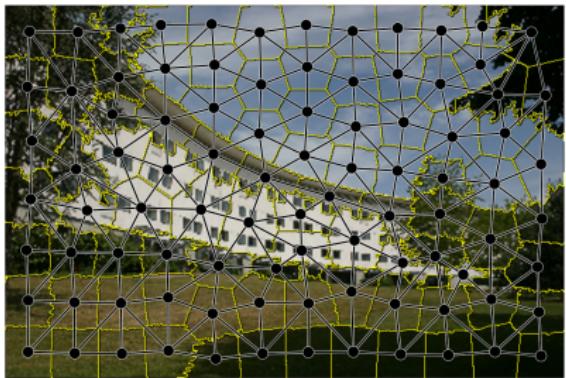


- Ground is at bottom.
- Walls are behind other objects.
- Ground is often in front of other objects.
- Furniture is in front of the walls.

Summary

- Can incorporate geometric relations into CRF.
- Learn all potentials.
- Exact learning of CRF possible.

How Intractable Are Loopy Models?



- In general, inference in loopy models is NP-hard.
- Learning relies on inference.
- How good can we get in practice?
- Focusing on 1-slack cutting plane algorithm.

Efficient Caching

$$\ell(x^i, y^i, w) = [\max_{y \in \mathcal{Y}} \Delta(y^i, y) + w^T \psi(x^i, y) - w^T \psi(x^i, y^i)]_+ \quad (1)$$

- Maximizing over y main bottleneck.

Efficient Caching

$$\ell(x^i, y^i, w) = [\max_{y \in \mathcal{Y}} \Delta(y^i, y) + w^T \psi(x^i, y) - w^T \psi(x^i, y^i)]_+ \quad (1)$$

- Maximizing over y main bottleneck.
- Reusing (caching) of previous found maxima can speed up optimization.

Efficient Caching

$$\ell(x^i, y^i, w) = [\max_{y \in \mathcal{Y}} \Delta(y^i, y) + w^T \psi(x^i, y) - w^T \psi(x^i, y^i)]_+ \quad (1)$$

- Maximizing over y main bottleneck.
- Reusing (caching) of previous found maxima can speed up optimization.
- Natural trade-off between time spent on finding maximum (inference) and time spent on optimization.

Efficient Caching

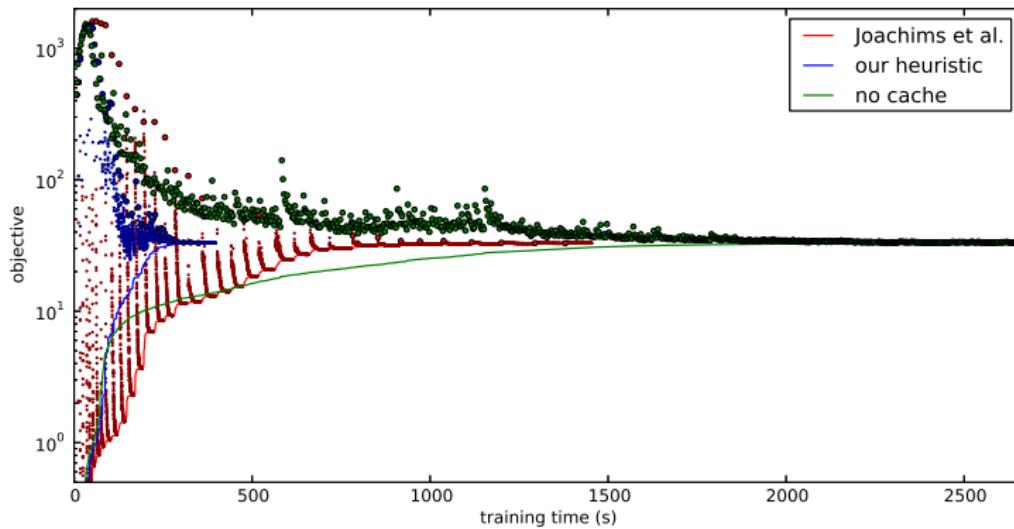
$$\ell(x^i, y^i, w) = [\max_{y \in \mathcal{Y}} \Delta(y^i, y) + w^T \psi(x^i, y) - w^T \psi(x^i, y^i)]_+ \quad (1)$$

- Maximizing over y main bottleneck.
- Reusing (caching) of previous found maxima can speed up optimization.
- Natural trade-off between time spent on finding maximum (inference) and time spent on optimization.
- Novel heuristic

$$o^C - o^I < \frac{1}{2}(o^I - o_W)$$

where o^C is the cached primal, o^I primal using inference I and o_W the dual.

Caching Comparison



Combining Inference Algorithms

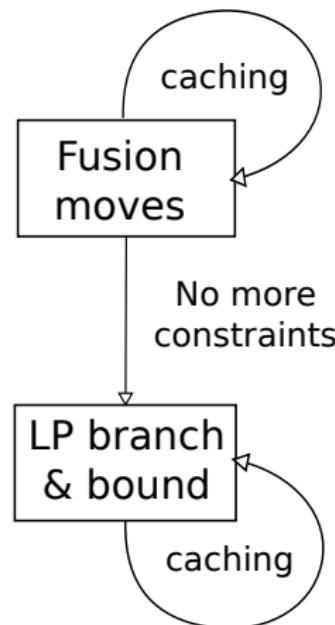
- Use fast inference in the beginning,
use good inference at the end.

Combining Inference Algorithms

- Use fast inference in the beginning,
use good inference at the end.
- Theoretically sound with cutting
plane algorithm.

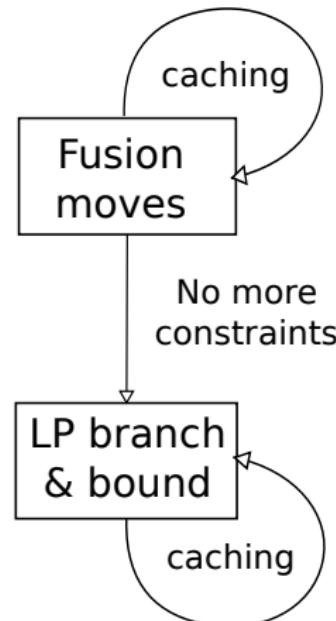
Combining Inference Algorithms

- Use fast inference in the beginning, use good inference at the end.
- Theoretically sound with cutting plane algorithm.
- Combine three procedures using efficient heuristic.

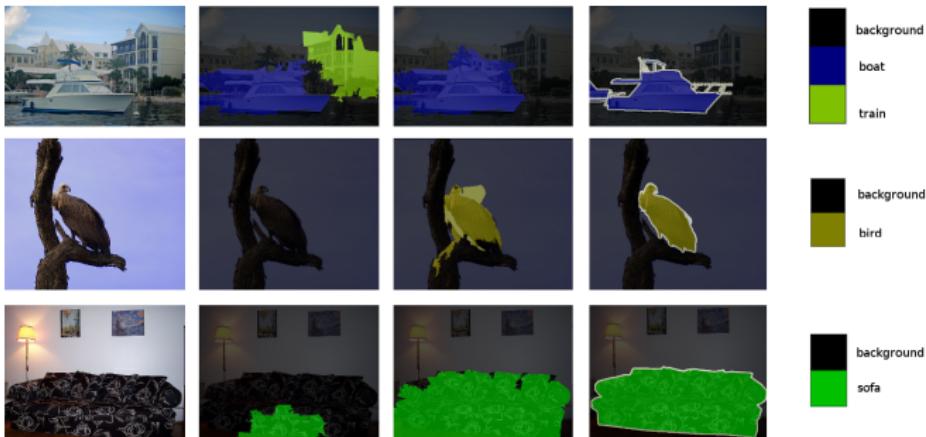


Combining Inference Algorithms

- Use fast inference in the beginning, use good inference at the end.
- Theoretically sound with cutting plane algorithm.
- Combine three procedures using efficient heuristic.
- Result: can learn exactly on two popular segmentation benchmarks!



Results on Pascal VOC2010 and MSRC-21



Pascal VOC 2010	Jaccard
Unary terms only	27.5
Pairwise model (move making)	30.2
Pairwise model (exact)	30.4
?	27.4
?	30.2
?	30.8

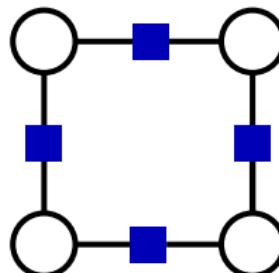
MSRC-21	Average	Global
Unary terms only	77.7	83.2
Pairwise model (move making)	79.6	84.6
Pairwise model (exact)	79.0	84.3
?	75.8	85.0
?	77	75
?	78.9	83.7

Summary

- We can learn some interesting loopy models exactly.
- Move-making was already very close.
- Exact learning is not necessarily slow (≈ 5 min).
- Makes learning more reproducible and simpler.

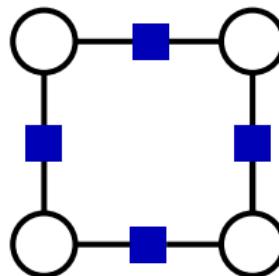
PyStruct - Simple Structured Prediction

- Implements common and state-of-the-art structural solvers (optimizing for w).



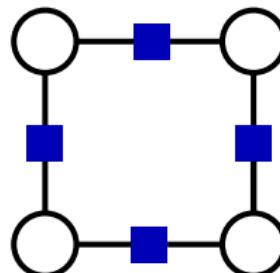
PyStruct - Simple Structured Prediction

- Implements common and state-of-the-art structural solvers (optimizing for w).
- Implements powerful models for many use cases.



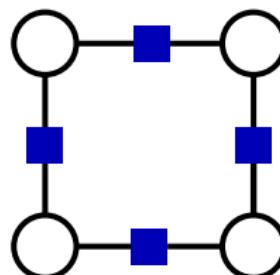
PyStruct - Simple Structured Prediction

- Implements common and state-of-the-art structural solvers (optimizing for w).
- Implements powerful models for many use cases.
- Interface to popular inference libraries: LibDAI, OpenGM, QPBO, AD3.

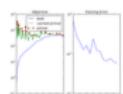


PyStruct - Simple Structured Prediction

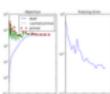
- Implements common and state-of-the-art structural solvers (optimizing for w).
- Implements powerful models for many use cases.
- Interface to popular inference libraries: LibDAI, OpenGM, QPBO, AD3.
- Well documented, includes common benchmarks.



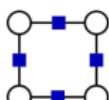
Examples



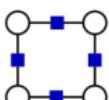
Plotting the objective and constraint caching in 1-slack SSVM



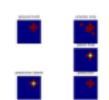
Efficient exact learning of 1-slack SSVMs



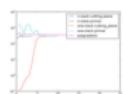
SVM as CRF



Semantic Image Segmentation on Pascal VOC



Latent Dynamics CRF



SVM objective values



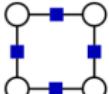
Learning directed interactions on a 2d grid



Learning interactions on a 2d grid



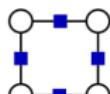
OCR Letter sequence recognition



Crammer-Singer Multi-Class SVM



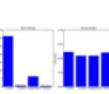
Latent SVM for odd vs. even digit classification



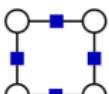
Mult-label classification



Latent Variable Hierarchical CRF



Binary SVM as SSVM



Comparing PyStruct and SVM-Struct

Summary of Contributions

- Loss-based learning yields meaningful geometric relations in RGB-D data. State-of-the-art results on NYU V2 RGB-D dataset.

Summary of Contributions

- Loss-based learning yields meaningful geometric relations in RGB-D data. State-of-the-art results on NYU V2 RGB-D dataset.
- Show feasibility of exact learning for image segmentation, state-of-the-art results on MSRC-21, on par with comparable approaches on Pascal VOC 2010.

Summary of Contributions

- Loss-based learning yields meaningful geometric relations in RGB-D data. State-of-the-art results on NYU V2 RGB-D dataset.
- Show feasibility of exact learning for image segmentation, state-of-the-art results on MSRC-21, on par with comparable approaches on Pascal VOC 2010.
- Demonstrate feasibility of weakly supervised object class segmentation.

Summary of Contributions

- Loss-based learning yields meaningful geometric relations in RGB-D data. State-of-the-art results on NYU V2 RGB-D dataset.
- Show feasibility of exact learning for image segmentation, state-of-the-art results on MSRC-21, on par with comparable approaches on Pascal VOC 2010.
- Demonstrate feasibility of weakly supervised object class segmentation.
- General purpose clustering algorithm, theoretically sound objective, efficient, no parameters, state-of-the art performance.

Summary of Contributions

- Loss-based learning yields meaningful geometric relations in RGB-D data. State-of-the-art results on NYU V2 RGB-D dataset.
- Show feasibility of exact learning for image segmentation, state-of-the-art results on MSRC-21, on par with comparable approaches on Pascal VOC 2010.
- Demonstrate feasibility of weakly supervised object class segmentation.
- General purpose clustering algorithm, theoretically sound objective, efficient, no parameters, state-of-the art performance.
- Well tested, efficient software implementation for learning structured prediction. Already used by other researchers.

Future Directions

- More complex models for semantic segmentation.
- Integrate over time in NYU dataset.
- ITM for superpixels.
- Large scale weakly supervised segmentation.

Overview

- *PyStruct - Structured Prediction in Python*
Andreas C. Müller and Sven Behnke. Journal of Machine Learning Research 2014.
- *Learning a Loopy Model for Semantic Segmentation Exactly*
Andreas C. Müller and Sven Behnke. International Conference on Computer Vision Theory and Applications 2014.
- *Learning Depth-Sensitive Conditional Random Fields for Semantic Segmentation*
Andreas C. Müller and Sven Behnke. International Conference on Robotics and Automation 2014.
- *Multi-Instance Methods for Partially Supervised Image Segmentation*
Andreas C. Müller and Sven Behnke. IARP Workshop on Partially Supervised Learning 2011.
- *Information Theoretic Clustering using Minimum Spanning Trees*
Andreas C. Müller, Sebastian Nowozin and Christoph H. Lampert. German Conference on Pattern Recognition 2012.

n -Slack Cutting Plane Training of Structural SVMs

Require: training samples $\{(x^1, y^1), \dots, (x^k, y^k)\}$, regularization parameter C , stopping tolerance ϵ .

Ensure: parameters θ , slack (ξ_1, \dots, ξ_k)

1: $\mathcal{W}_i \leftarrow \emptyset, \xi_i \leftarrow 0$ for $i = 1, \dots, k$

2: **repeat**

3: **for** $i=1, \dots, k$ **do**

4: $\hat{y} \leftarrow l(x^i, y^i, \theta) := \arg \max_{\hat{y} \in \mathcal{Y}} \delta(y^i, \hat{y}) - \theta^T [\Phi(x^i, y^i) - \Phi(x^i, \hat{y})]$

5: **if** $\delta(y^i, \hat{y}) - \theta^T [\Phi(x^i, y^i) - \Phi(x^i, \hat{y})] \geq \xi_i + \epsilon$ **then**

6: $\mathcal{W}_i \leftarrow \mathcal{W}_i \cup \{\hat{y}\}$

7:

$$(\theta, \xi_1, \dots, \xi_k) \leftarrow \arg \min_{\theta, \xi_1, \dots, \xi_k} \frac{\|\theta\|^2}{2} + C \sum_{i=1}^k \xi_i$$

s.t. for $i = 1, \dots, k$ $\forall \hat{y} \in \mathcal{W}_i :$

$$\theta^T [\Phi(x^i, y^i) - \Phi(x^i, \hat{y}^i)] \geq \delta(y^i, \hat{y}^i) - \xi_i$$

8: **until** no \mathcal{W}_i changes anymore.

1-Slack Cutting Plane Training of Structural SVMs

Require: training samples $\{(x^i, y^i), \dots, (x^i, y^i)\}$, regularization parameter C , stopping tolerance ϵ .

Ensure: parameters θ , slack ξ

1: $\mathcal{W} \leftarrow \emptyset$

2: **repeat**

3:

$$(\theta, \xi) \leftarrow \arg \min_{\theta, \xi} \frac{\|\theta\|^2}{2} + C\xi$$

s.t. $\forall \hat{\mathbf{y}} = (\hat{y}^1, \dots, \hat{y}^k) \in \mathcal{W}$:

$$\theta^T \sum_{i=1}^k [\Phi(x^i, y^i) - \Phi(x^i, \hat{y}^i)] \geq \sum_{i=1}^k \delta(y^i, \hat{y}^i) - \xi$$

4: **for** $i=1, \dots, k$ **do**

$$5: \quad \hat{y}^i \leftarrow l(x^i, y^i, \theta) := \arg \max_{\hat{y} \in \mathcal{Y}} \sum_{i=1}^k \delta(y^i, \hat{y}) - \theta^T \sum_{i=1}^k [\Phi(x^i, y^i) - \Phi(x^i, \hat{y})]$$

6: $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\hat{y}^i, \dots, \hat{y}^i)\}$

$$7: \quad \xi' \leftarrow \sum_{i=1}^k \delta(y^i, \hat{y}^i) - \theta^T \sum_{i=1}^k [\Phi(x^i, y^i) - \Phi(x^i, \hat{y}^i)]$$

8: **until** $\| \xi' - \xi \| < \epsilon$

Block-Coordinate Frank-Wolfe Algorithm

Require: training samples $\{(x^i, y^i), \dots, (x^i, y^i)\}$, regularization parameter C , stopping tolerance ϵ .

Ensure: parameters θ

1: $\theta_0, \theta_0^j, \bar{\theta}_0 \leftarrow \mathbf{0}, \ell_0, \ell_0^j, t \leftarrow 0$

2: **repeat**

3: $t \leftarrow t + 1$

4: Pick i uniformly at random from $\{1, \dots, k\}$.

5: Perform loss-augmented prediction on sample i :

$$\hat{y} \leftarrow l(x^i, y^i, \theta) := \arg \max_{\hat{y} \in \mathcal{Y}} \Delta(y^i, \hat{y}) - \theta^T [\Phi(x^i, y^i) - \Phi(x^i, \hat{y})]$$

6: Compute parameter and loss updates based on sample i :

$$\theta_s \leftarrow \frac{C}{n} \Phi(x, \hat{y})$$

$$\ell_s \leftarrow \frac{C}{n} \delta(y^i, \hat{y})$$

7: Compute optimum step size η :

$$\eta \leftarrow \frac{(\theta_t^j - \theta_s)^T \theta_t + C(\ell_s - \ell_k^j)}{\|\theta_t^j - \theta_s\|^2} \text{ and clip to } [0, 1]$$

8: Update per-sample parameters and loss estimate:

$$\theta_{t+1}^j \leftarrow (1 - \eta) \theta_{t+1}^j + \eta \theta_s$$

$$\ell_{t+1}^j \leftarrow (1 - \eta) \theta_{t+1}^j + \eta \ell_s$$

9: Update global parameters and loss estimate:

$$\theta_{t+1} \leftarrow \theta_{t+1} + \theta_t^j - \theta_{t+1}^j$$

$$\ell_{t+1} \leftarrow \ell_{t+1} + \ell_t^j - \ell_{t+1}^j$$

10: Compute the weighted running average:

$$\bar{\theta}_{t+1} = \frac{k}{k+2} \bar{\theta}_k + \frac{2}{k+2} \theta_{k+1}$$

11: **until** $(\theta - \theta_s)^T \theta - \ell + \ell_s \leq \epsilon$