Structured Prediction for Semantic Segmentation

Andreas C. Müller

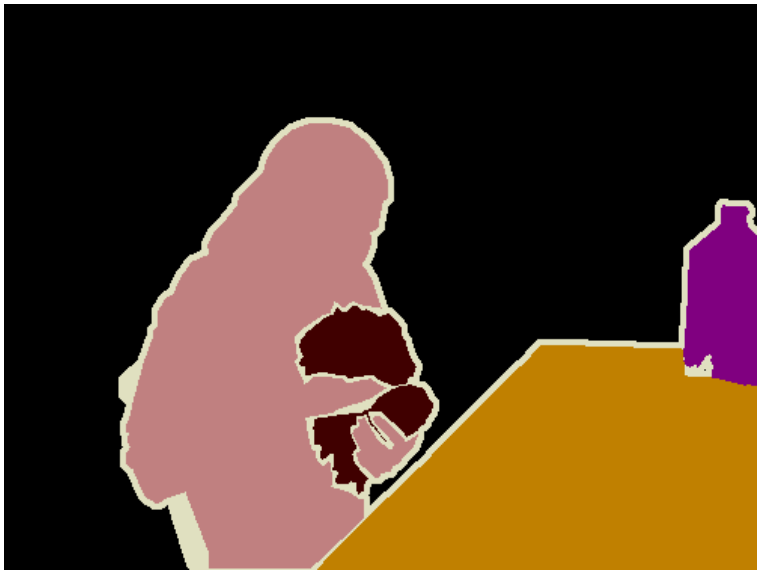
Columbia University

April 10, 2018

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

Semantic Segmentation

Semantic Segmentation



Multi-Label Classification

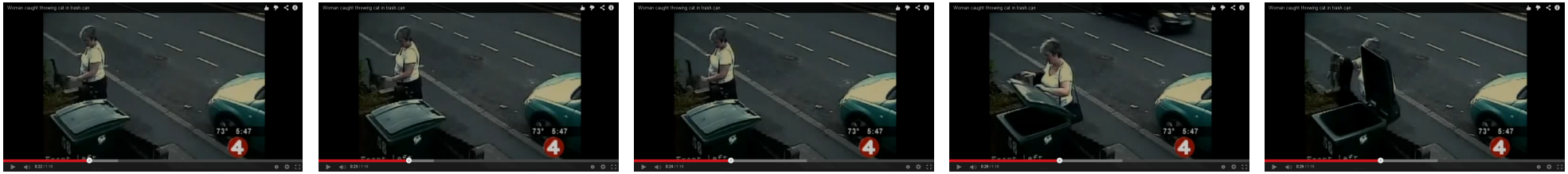
	Politics	Sports	Finance	Domestic	Religion
News Story1	1	0	0	1	1
News Story2	0	1	0	1	0
News Story3	0	0	1	0	0

Multi-Label Classification

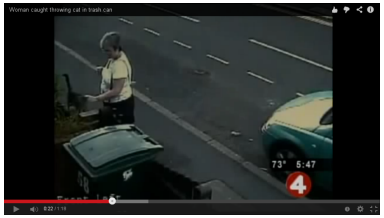
	Politics	Sports	Finance	Domestic	Religion
News Story1	1	0	0	1	1
News Story2	0	1	0	1	0
News Story3	0	0	1	0	0

	Owns Car	Smokes	Married	Self-Employed	Has Kids
Customer1	1	0	1	0	1
Customer2	1	1	0	1	0
Customer3	0	1	1	0	0

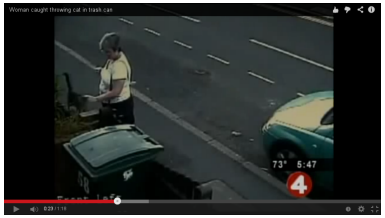
Sequence Tagging



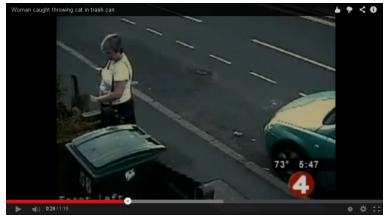
Sequence Tagging



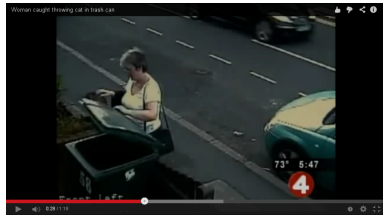
Stroke cat.



Stroke cat.



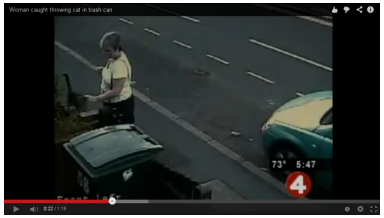
Stroke cat.



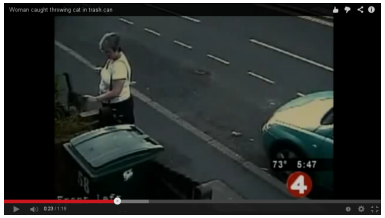
Open trash can.

Put cat in trash can.

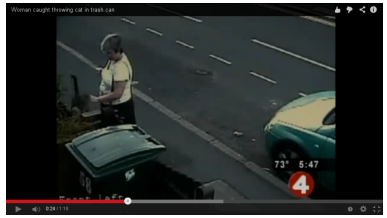
Sequence Tagging



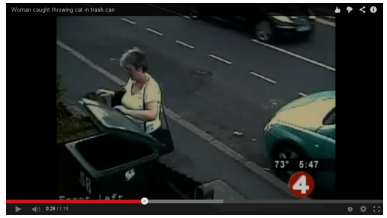
Stroke cat.



Stroke cat.

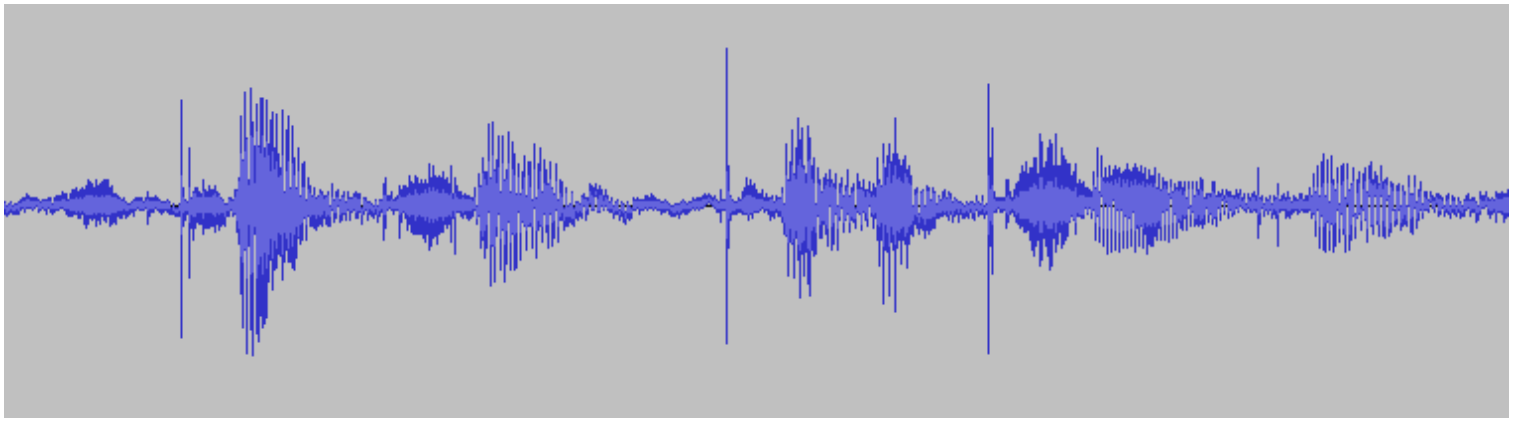


Stroke cat.

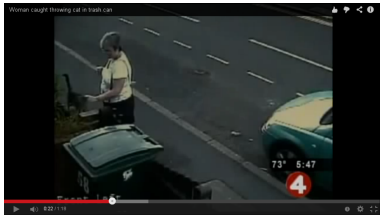


Open trash can.

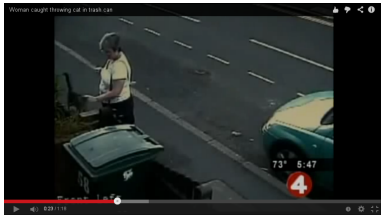
Put cat in trash can.



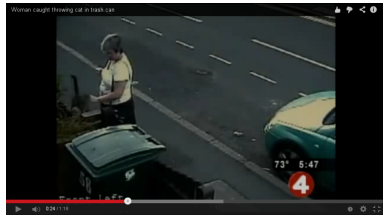
Sequence Tagging



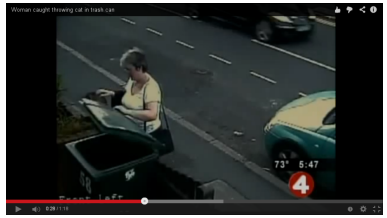
Stroke cat.



Stroke cat.

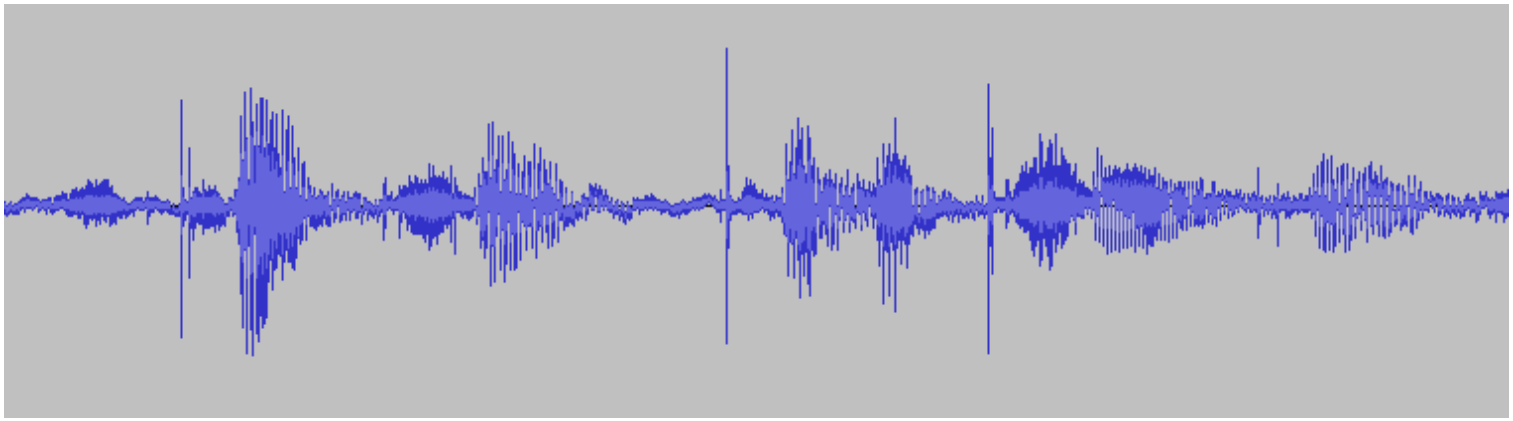


Stroke cat.



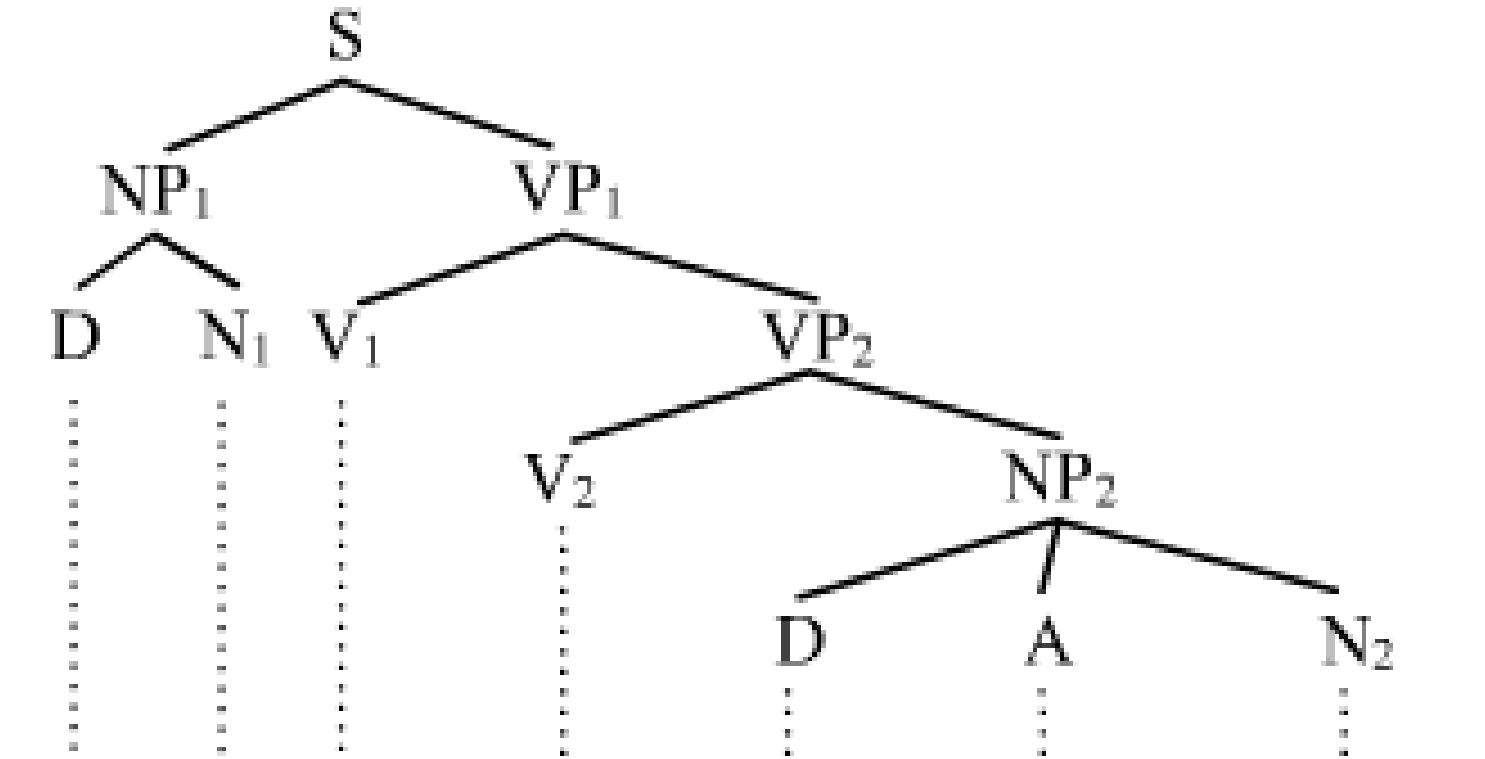
Open trash can.

Put cat in trash can.



Struc-tured pre-dic-tion in Py-thon

Predicting Parse Trees



Constituency relation (PSG)

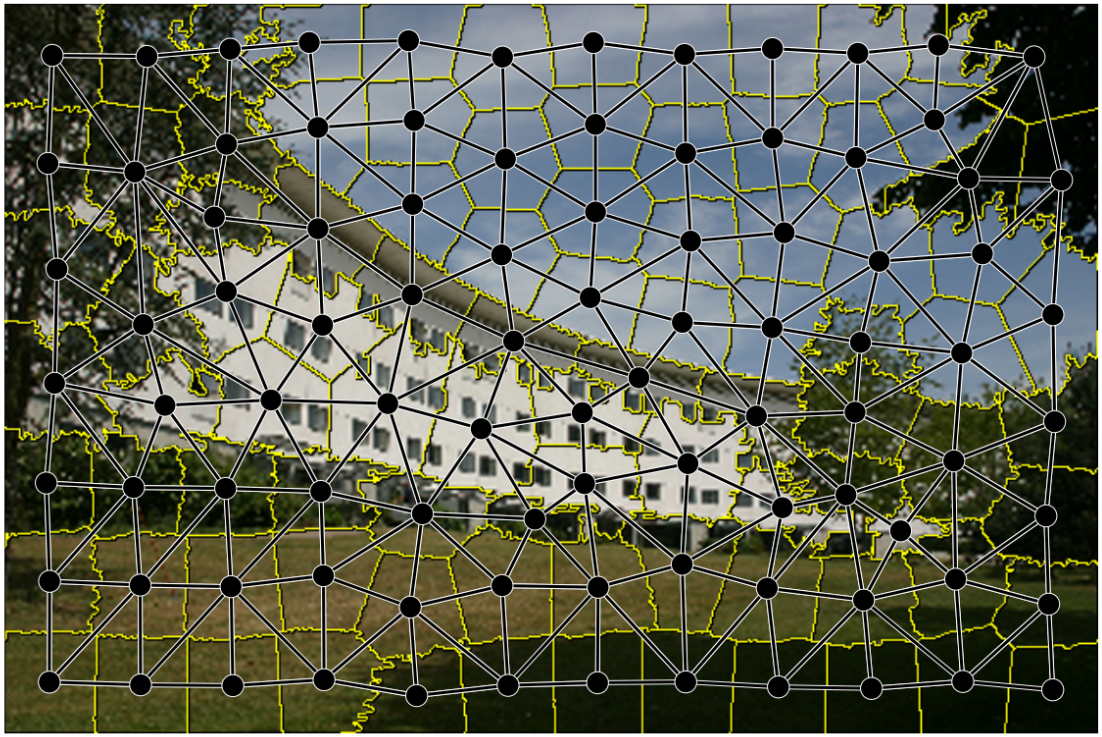
Structured Prediction for Semantic Segmentation

Semantic Segmentation as Structured Prediction

Semantic Segmentation as Structured Prediction

Semantic Segmentation as Structured Prediction

Semantic Segmentation as Structured Prediction



Predicting Structured Objects

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

Predicting Structured Objects

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

If you like:

$$\arg\max_{y \in \mathcal{Y}} p(y|x, w)$$

Predicting Structured Objects

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

If you like:

$$\arg\max_{y \in \mathcal{Y}} p(y|x, w)$$

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

Predicting discrete vectors

$$y = (y_1, y_2, \ldots, y_{n_i})$$

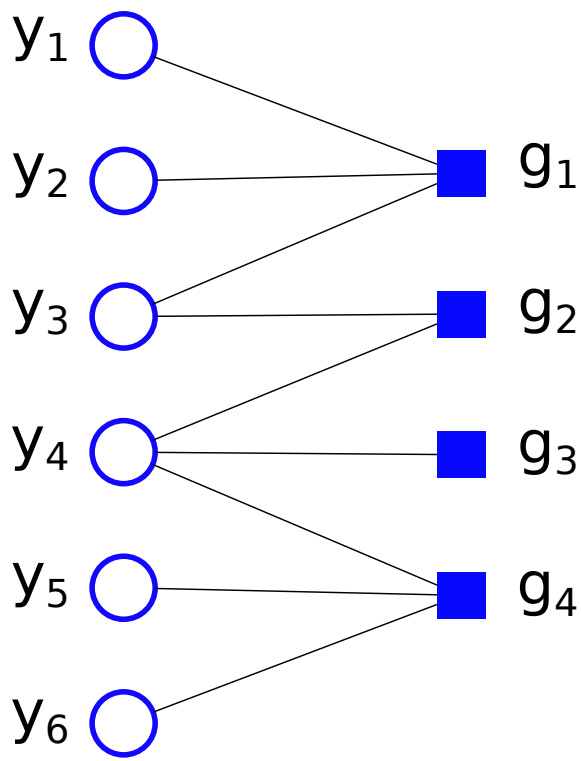
Predicting discrete vectors

$$y = (y_1, y_2, \ldots, y_{n_i})$$

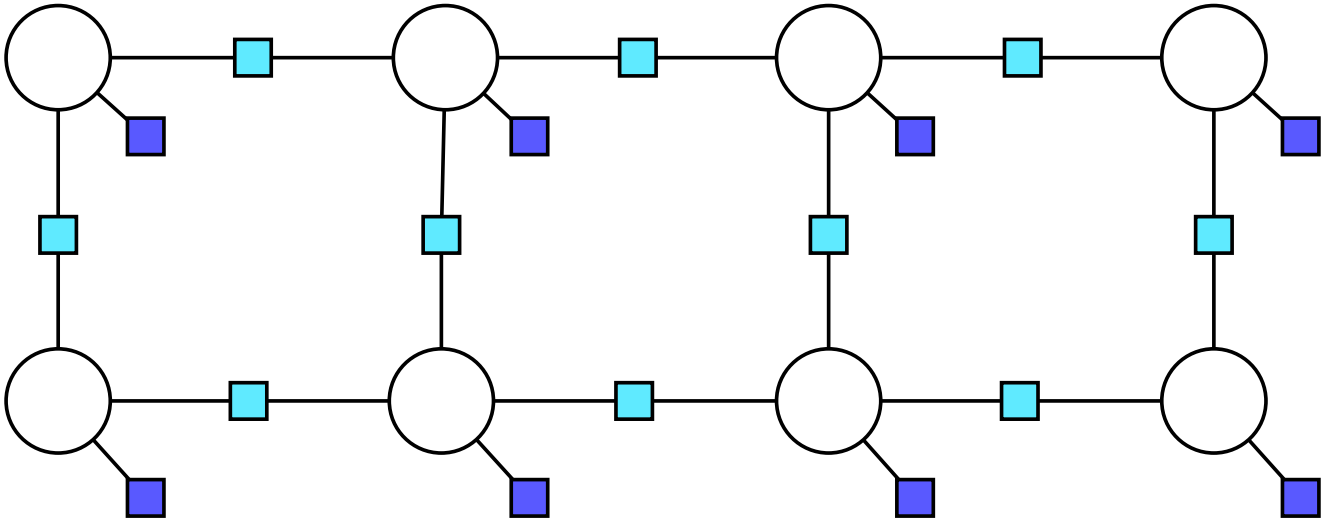
$$\begin{aligned} f(x, w) &= \underset{y \in \mathcal{Y}}{\arg\max}\, w^T \psi(x, y) \\ &= \underset{y_1, y_2, \ldots, y_{n_i}}{\arg\max}\, w^T \psi(x, y) \end{aligned}$$

Factor Graphs

$$g(x, y) = g_1(x, y_1, y_2, y_3) + g_2(x, y_3, y_4) + g_3(x, y_4) + g_4(x, y_4, y_5, y_6)$$

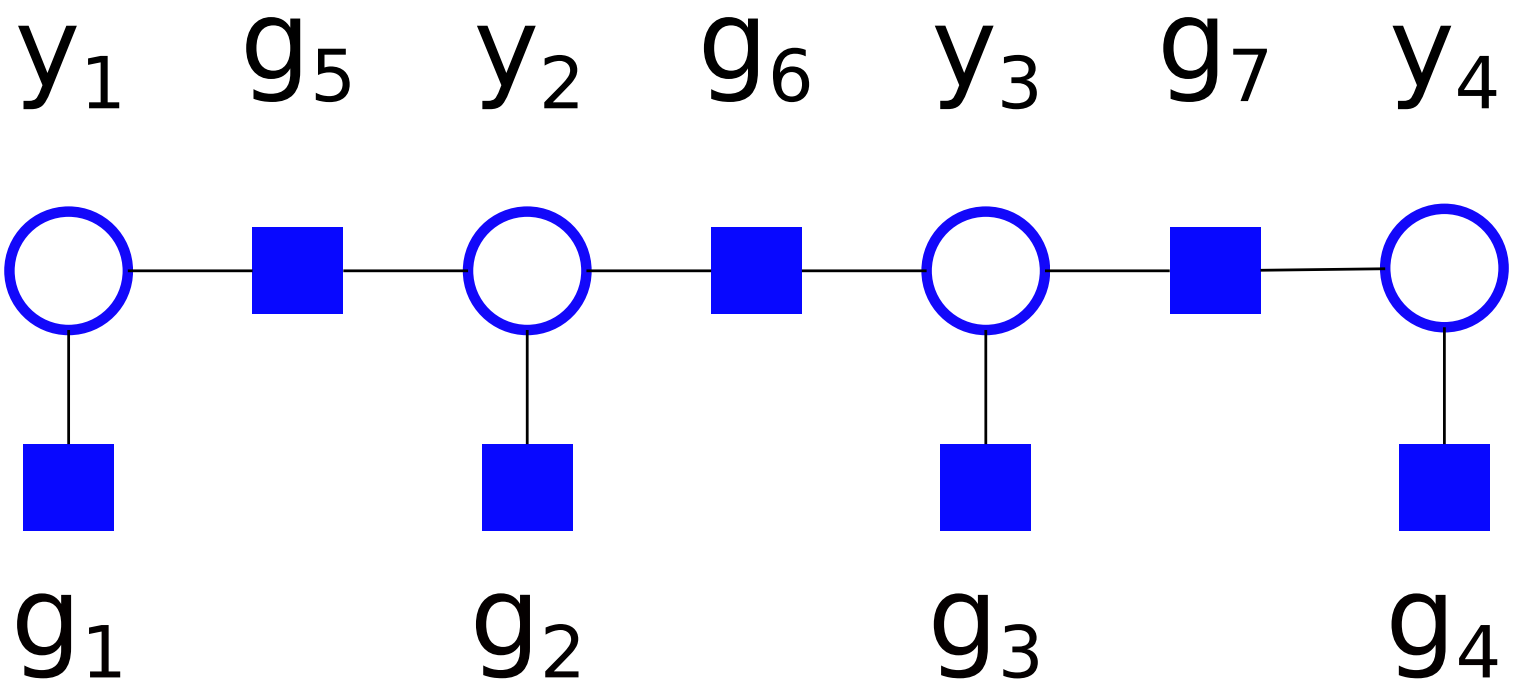


Pairwise Models



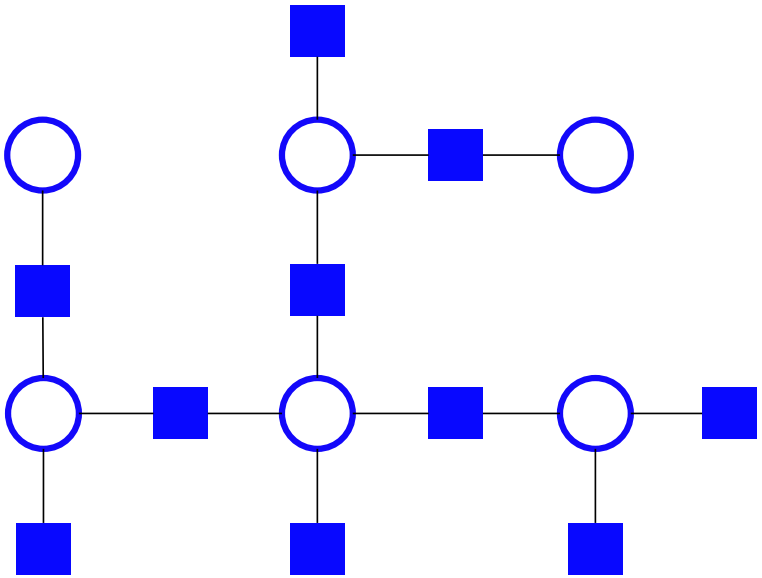
$$w^T \psi(x, y) = \sum_{(i,j) \in E} w_{i,j} \psi_{i,j}(x, y_i, y_j) + \sum_{i \in I} w_i \psi_i(x, y_i)$$

Factor Graph for HMM

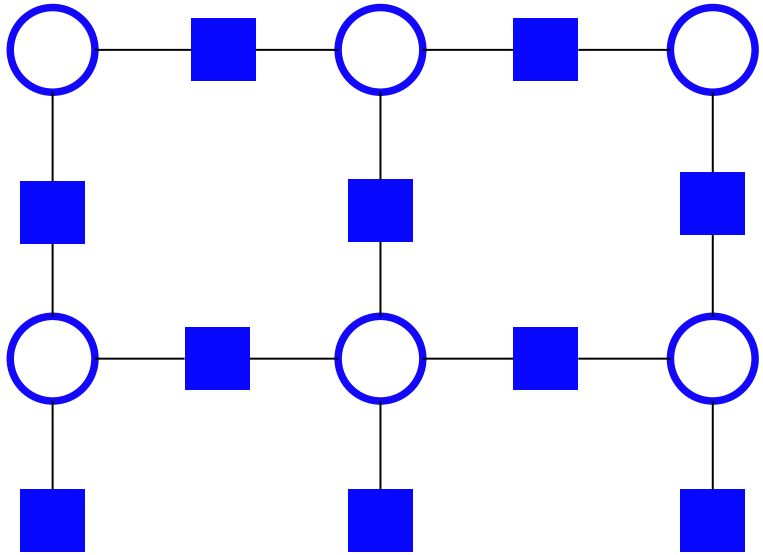


Inference

Easy



Tricky



Probabilistic Learning

$$p(y|x, w) = \frac{1}{Z} exp(w^T \psi(x, y))$$

$$Z = \sum_{y' \in \mathcal{Y}} exp(w^T \psi(x, y'))$$

Probabilistic Learning

$$p(y|x, w) = \frac{1}{Z} exp(w^T \psi(x, y))$$

$$Z = \sum_{y' \in \mathcal{Y}} exp(w^T \psi(x, y'))$$

Objective

$$\max_w \sum_i \log(p(y^i|x^i, w))$$

$$= \max_w \sum_i w^T \psi(x^i, y^i) - \log(Z)$$

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

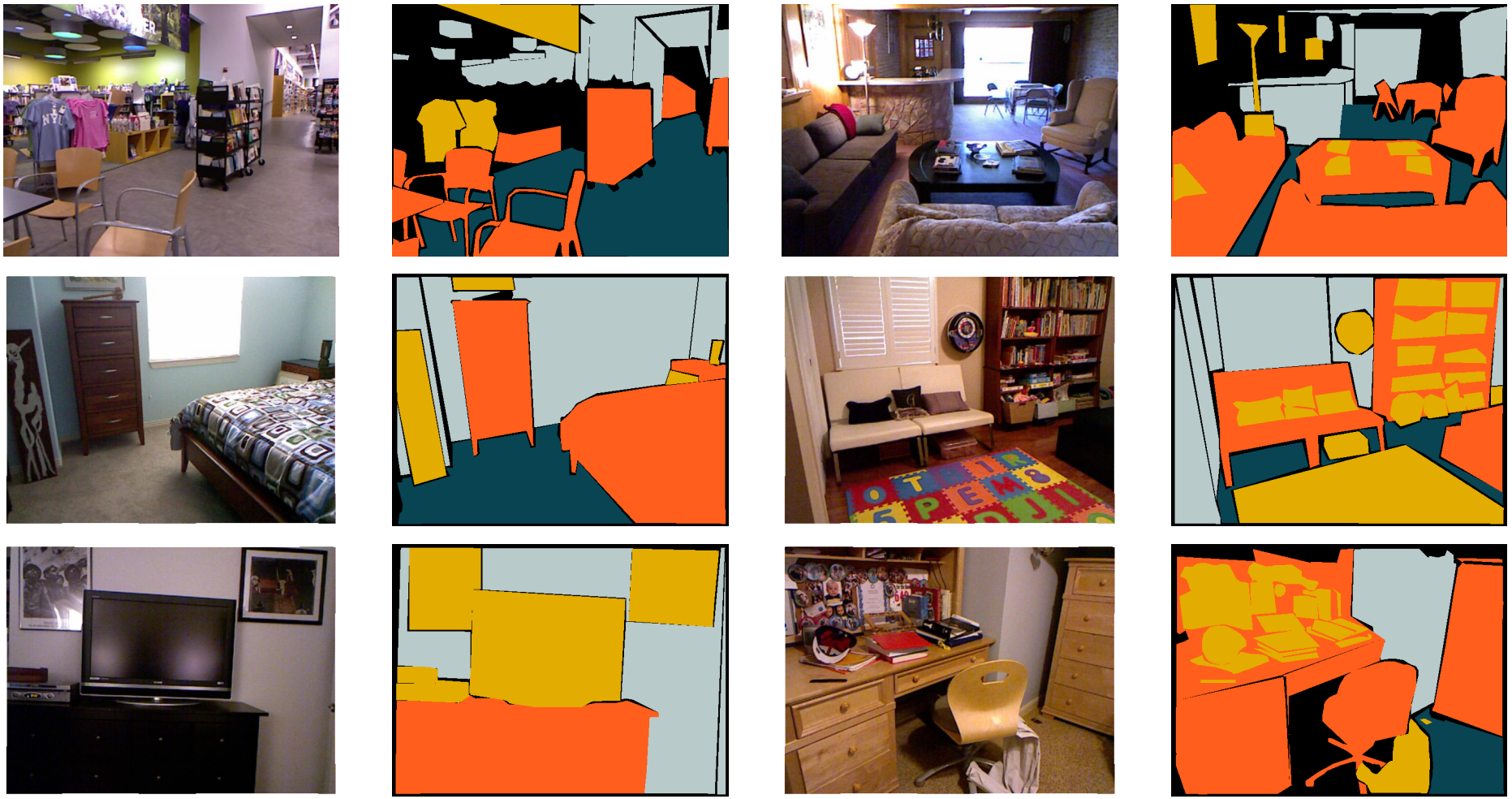
$$\min_w \frac{1}{2}\|w\|^2 + C \sum_i \ell(x^i, y^i, w)$$

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

(author?) [4]

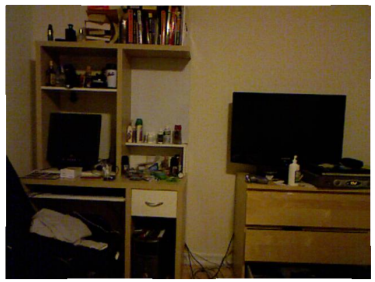
Learning Depth-Sensitive Conditional Random Fields

Dataset: NYUv2



795 training images, 654 test images.

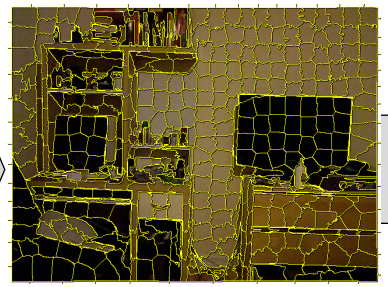
Overview



RGB Image

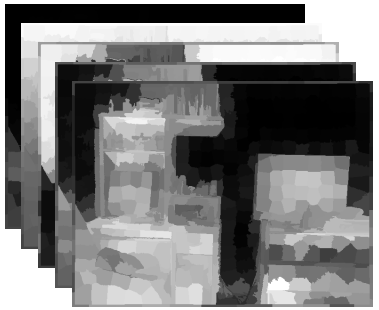
Depth Image

Random Forest



Superpixel

3D Point Cloud



Unary Features

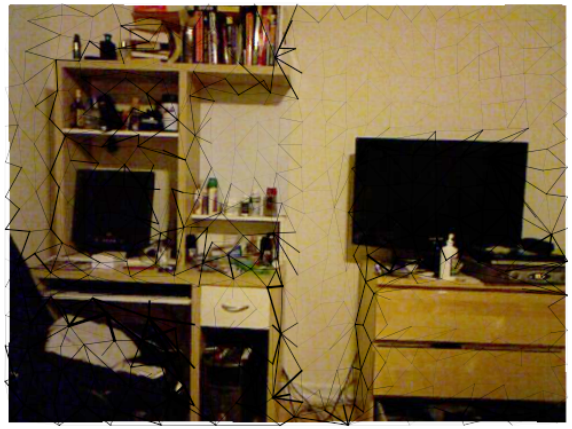
Pairwise Features

CRF Prediction

Pairwise Features

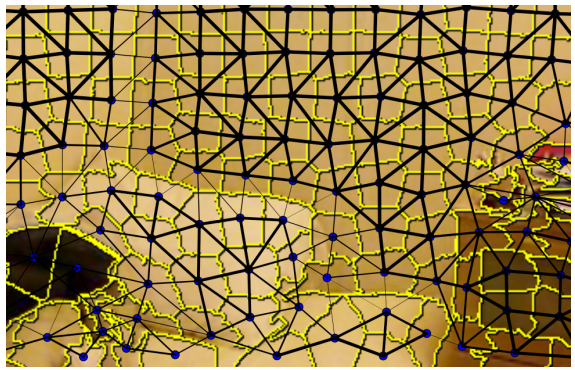
$$f_{\text{color}}(x_i, x_j) = \exp\left(-\gamma \|c_i - c_j\|^2\right)$$

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



$$f_{\text{direction}}(x_i, x_j) = \sphericalangle(\text{pos}_{x_i} - \text{pos}_{x_j}, [0, 1]^T)$$

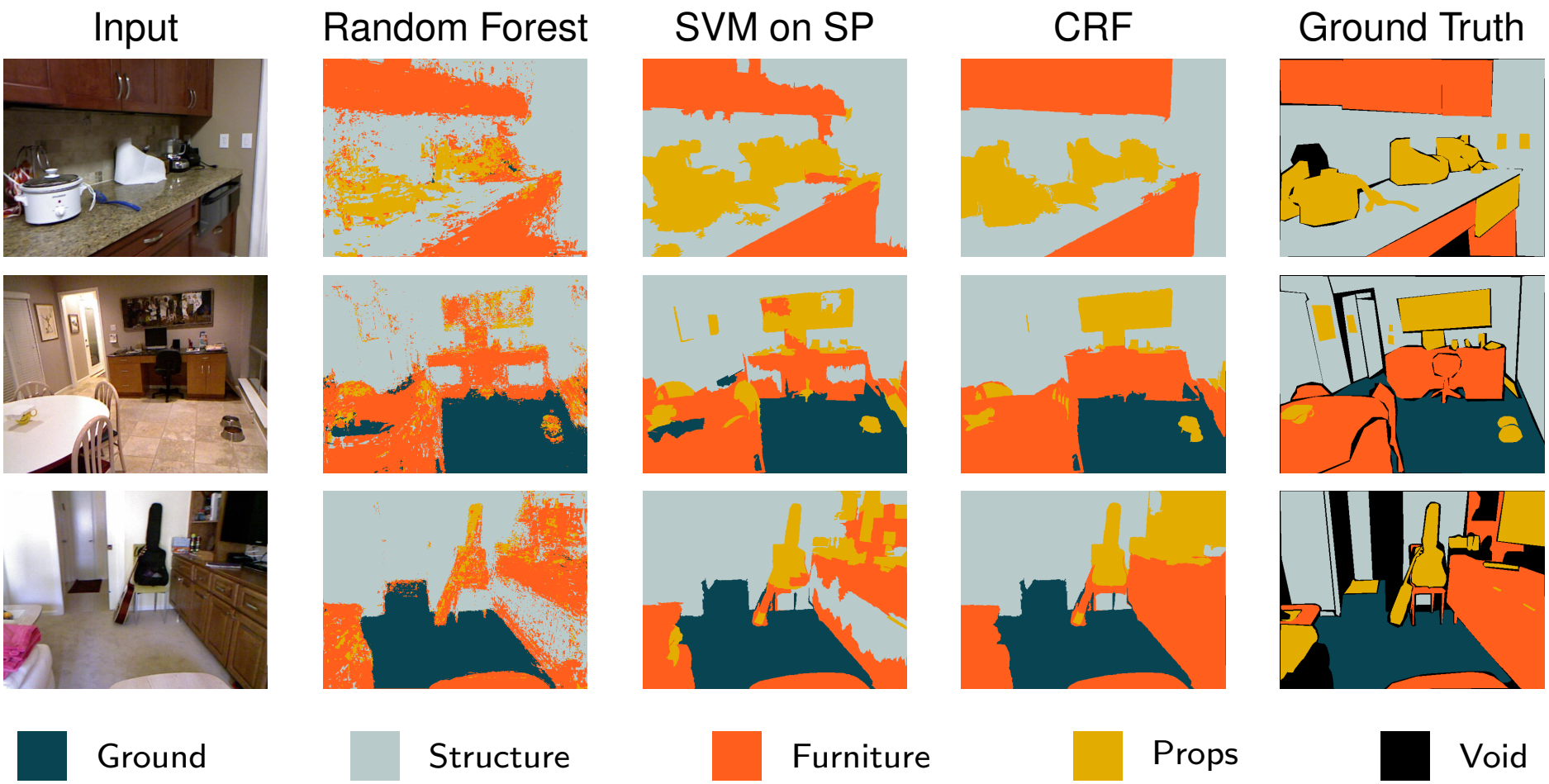
$$f_{\text{normals}}(x_i, x_j) = 1 - \frac{1}{\pi}\sphericalangle(\mathbf{n}_{x_i}, \mathbf{n}_{x_j})$$



Learning and Optimization

- 1-slack SSVM
- Inference using fusion moves and AD^3.
- Exact learning in loopy model [9].

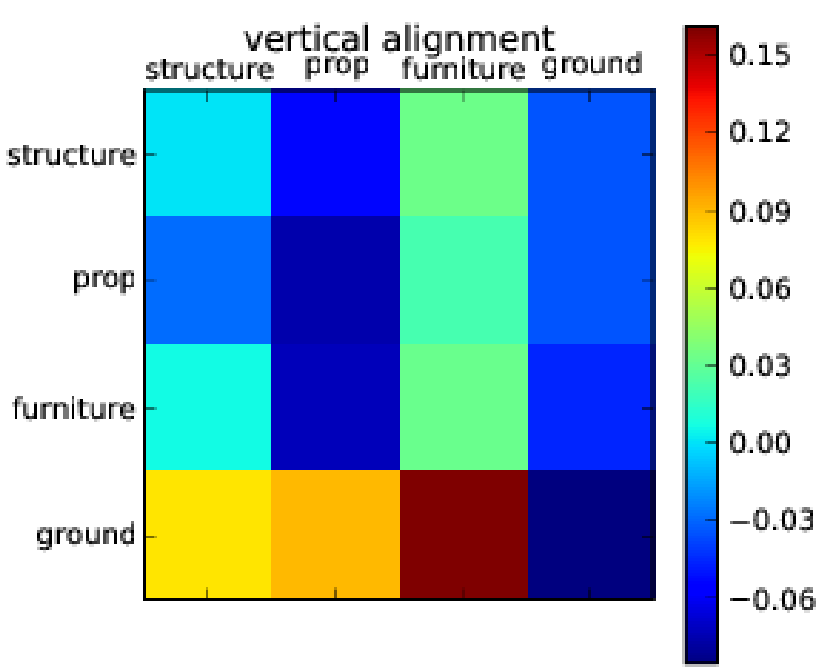
Qualitative Results

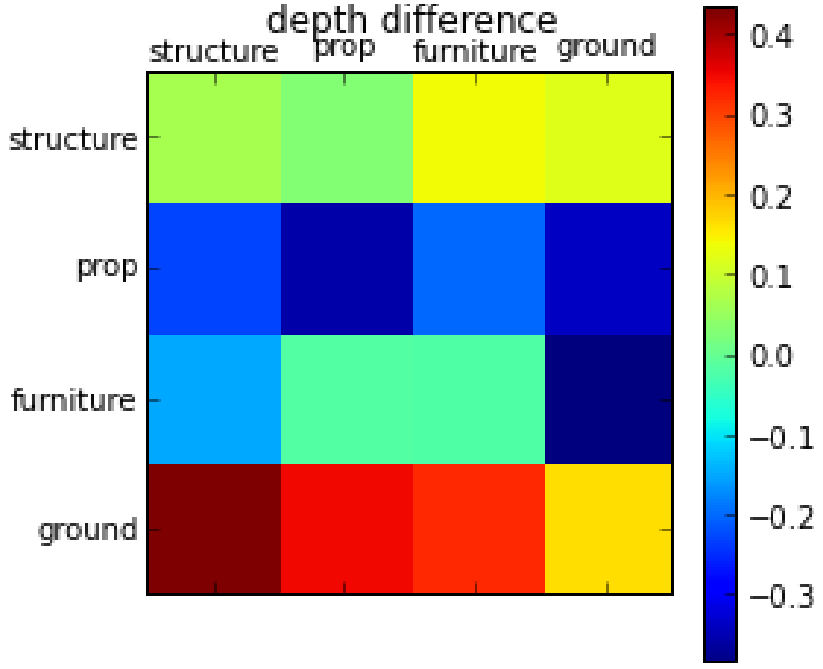


Results

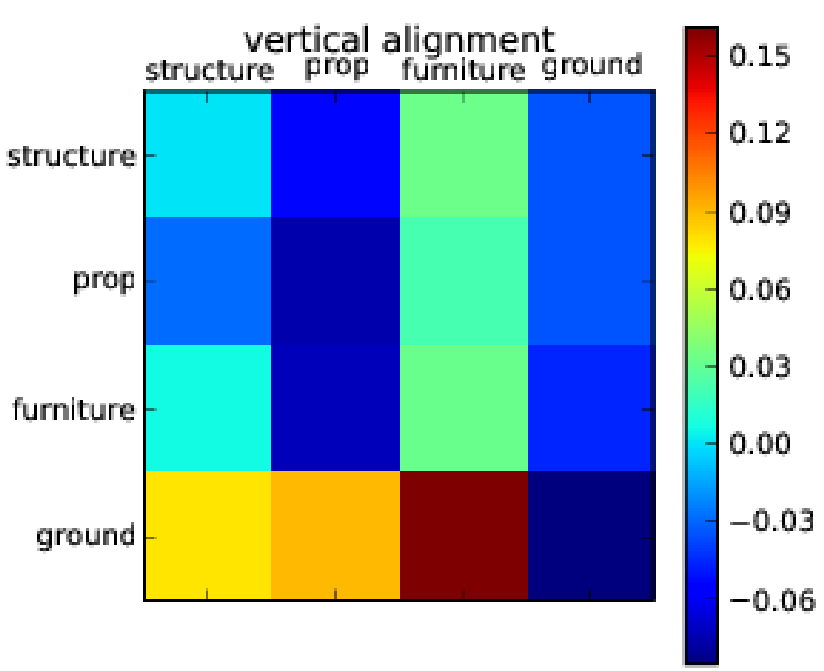
	ground	structure	furniture	props	class avg	pixel avg
RF	90.8	81.6	67.9	19.9	65.0	68.3
RF + SP	92.5	83.3	**73.8**	13.9	65.7	70.1
RF + SP + SVM	94.4	79.1	64.2	**44.0**	70.4	70.3
RF + SP + CRF	**94.9**	78.9	71.1	42.7	**71.9**	**72.3**
(author?) [10]	68	59	70	42	59.6	58.6
(author?) [1]	87.3	**86.1**	45.3	35.5	63.5	64.5
(author?) [11]†	**95.6**	83.0	**75.1**	14.2	67.0	70.9

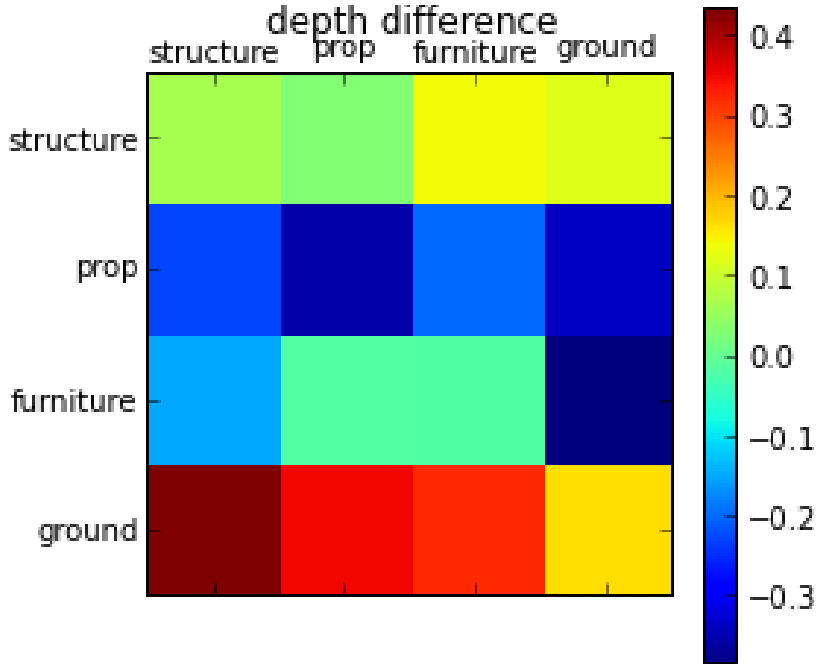
Learned Potentials





Learned Potentials





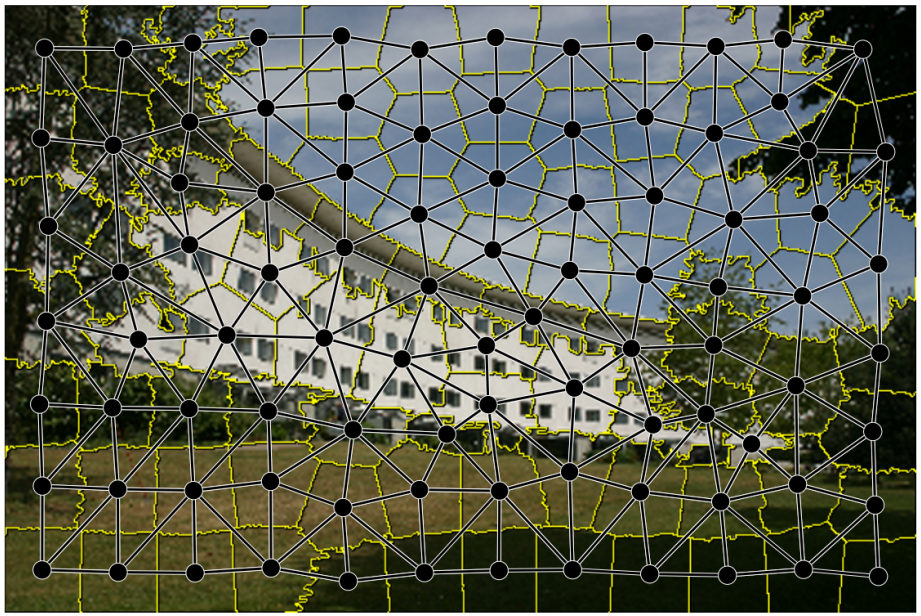
- Ground is at bottom.
- Walls are behind other objects.
- Ground is often in front of other objects.
- Furniture is in front of the walls.

Summary

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

Learning Loopy CRFs Exactly

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

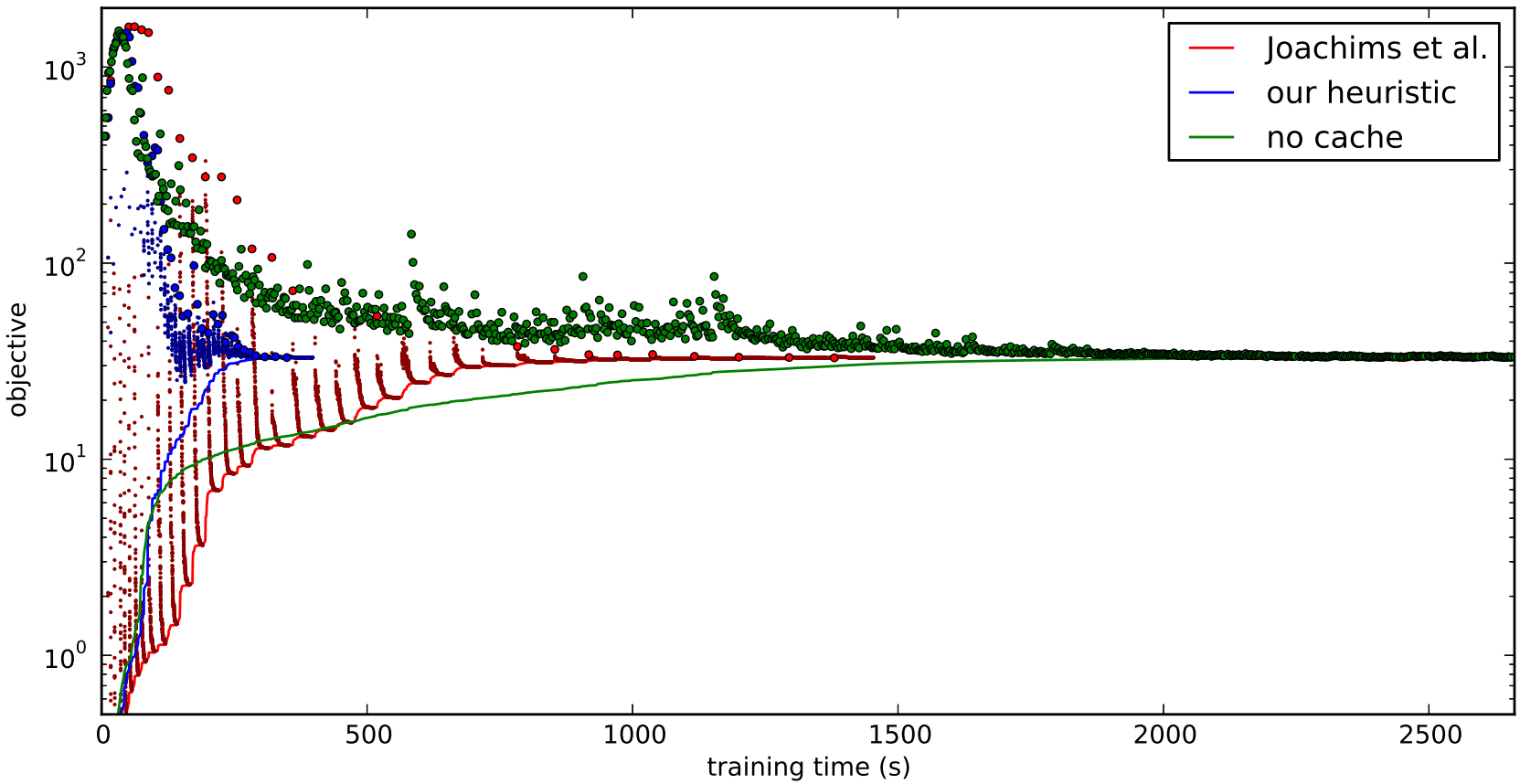
$$\ell(x^i, y^i, w) = [\max_{y \in \mathcal{Y}} \Delta(y^i, y) + w^T \psi(x^i, y) - w^T \psi(x^i, y^i)]_+ \quad (1)$$

- 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

$$o^C - o^I < \frac{1}{2}(o^I - o_{\mathcal{W}})$$

where o^C is the cached primal, o^I primal using inference I and $o_{\mathcal{W}}$ the dual.

Caching Comparison



Combining Inference Algorithms

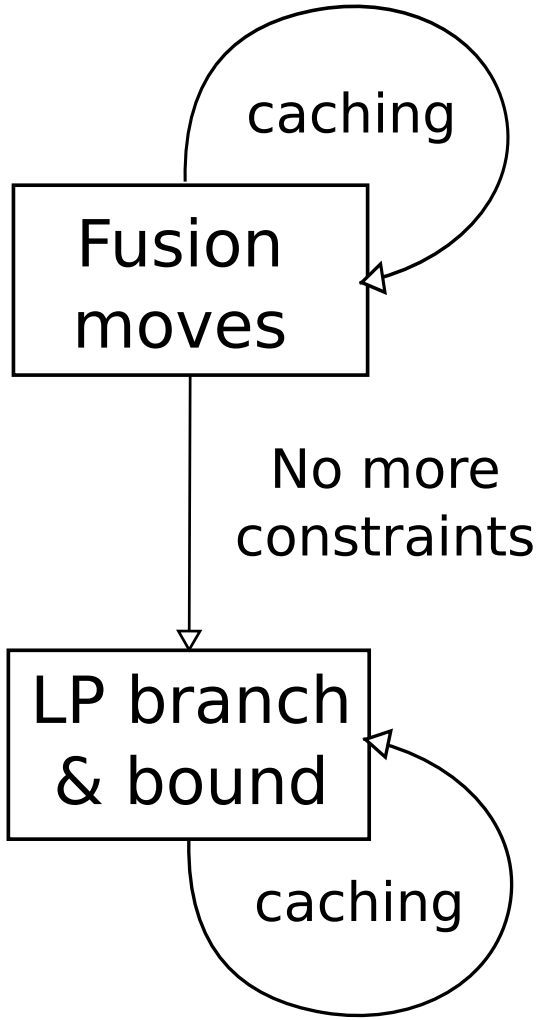
- Use fast inference in the beginning, use good inference at the end.

Combining Inference Algorithms

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

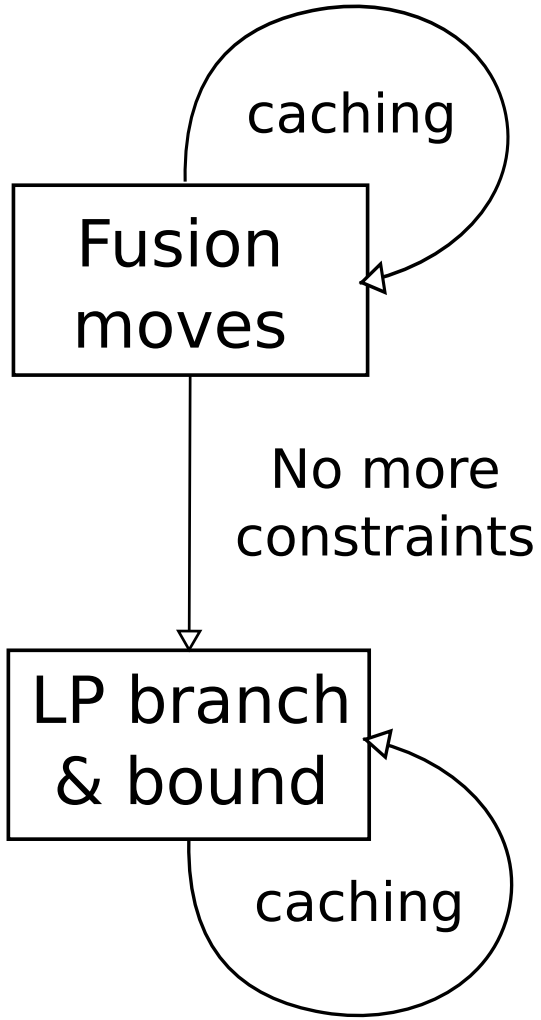
Combining Inference Algorithms

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

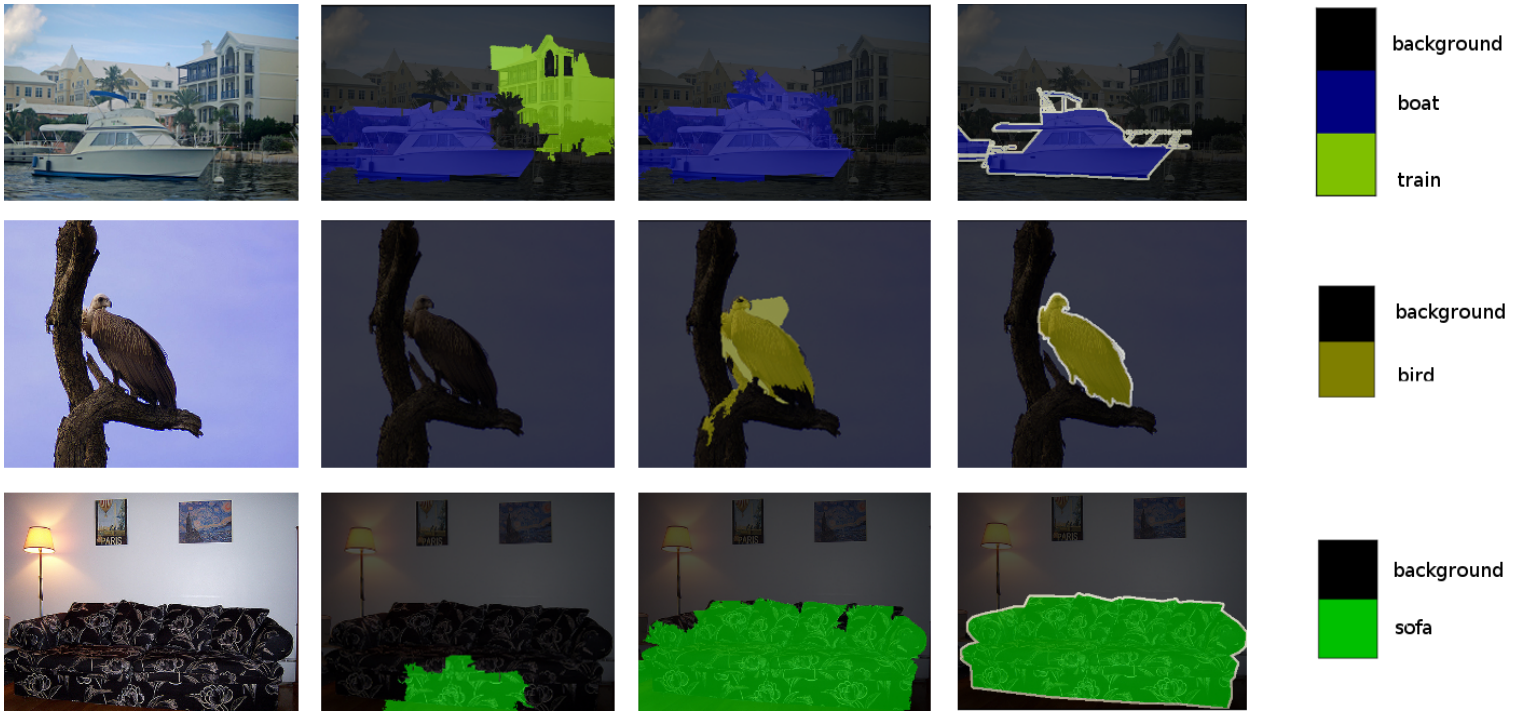


Combining Inference Algorithms

- 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
(author?) [2]	27.4
(author?) [5]	30.2
(author?) [6]	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
(author?) [7]	75.8	85.0
(author?) [3]	77	75
(author?) [8]	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

Estimator = Learner + Model + Inference

Simple structured prediction

Estimator = Learner + Model + Inference

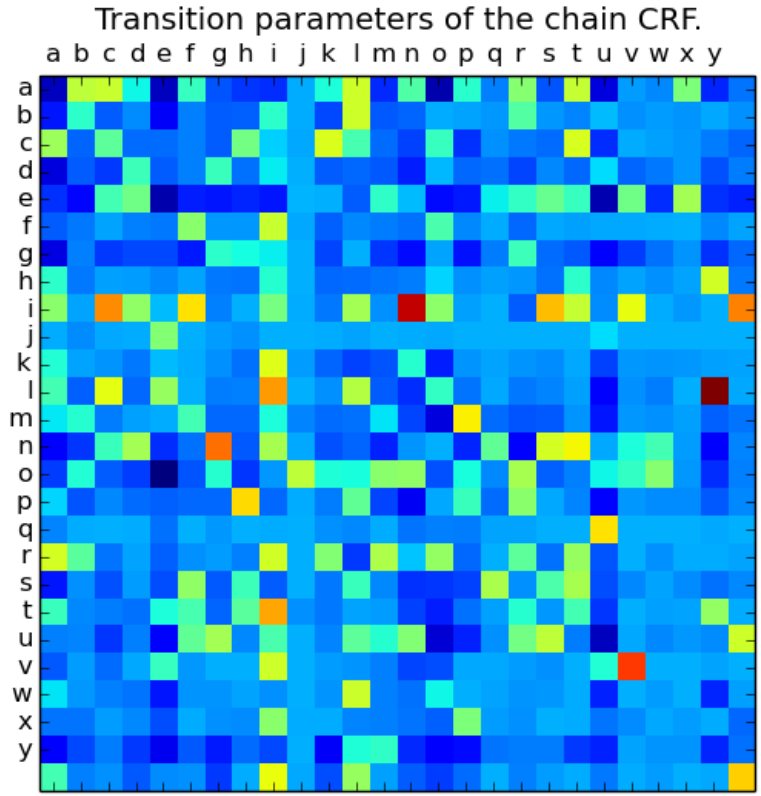
- Learner: SubgradientSSVM, StructuredPerceptron, OneSlackSSVM, LatentSSVM
- Model: BinaryClf, MultiLabelClf, ChainCRF, GraphCRF, EdgeFeatureGraphCRF
- Inference: Linear Programming, QPBO (PyQPBO), Dual Decomposition (AD3), Message Passing (OpenGM), Everything (OpenGM)

Example OCR

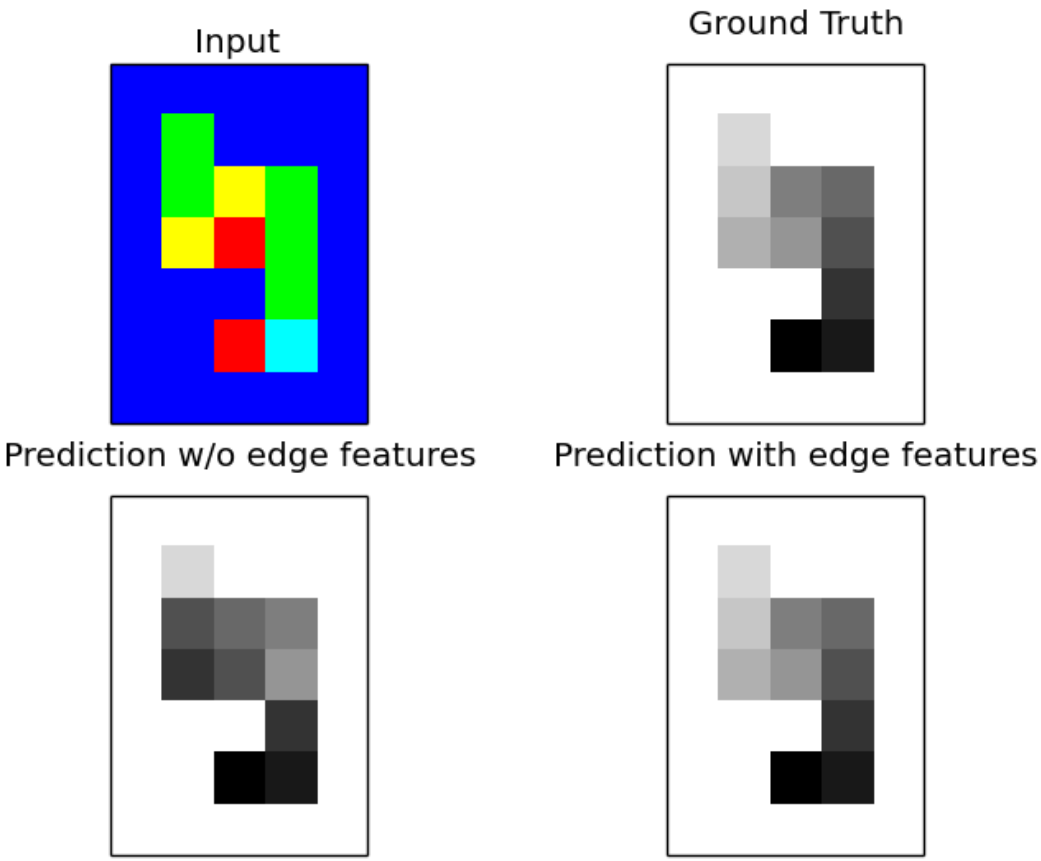
```
from pystruct.datasets import load_letters
from pystruct.models import ChainCRF
from pystruct.learners import OneSlackSSVM

abc = "abcdefghijklmnopqrstuvwxyz"

letters = load_letters()
X, y, folds = letters['data'], letters['labels'], letters['folds']
# we convert the lists to object arrays, as that makes slicing much more
# convenient
X, y = np.array(X), np.array(y)
X_train, X_test = X[folds == 1], X[folds != 1]
y_train, y_test = y[folds == 1], y[folds != 1]

# Train linear SVM
svm = LinearSVC(dual=False, C=.1)
# flatten input
svm.fit(np.vstack(X_train), np.hstack(y_train))

# Train linear chain CRF
model = ChainCRF()
ssvm = OneSlackSSVM(model=model, C=.1, inference_cache=50, tol=0.1, verbose=3)
ssvm.fit(X_train, y_train)
```

Example OCR


a f e t e r i a
a a f t a a f t m a r r i i a a
m b r a c e s
m m b b r r a a c c e e s s
p o i l f n g
p p a a i i c c i i m n d g
a f e t e r i a
a a f f o o d d o o l l z i n n

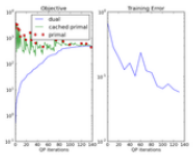

Transition parameters of the chain CRF.
a b c d e f g h i j k l m n o p q r s t u v w x y

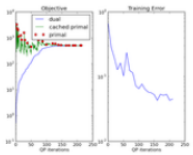
Example Snake

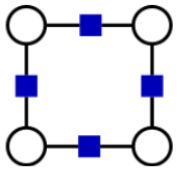


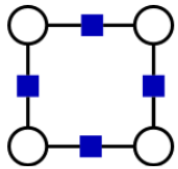
PyStruct | Introduction | Examples | References | Search

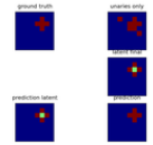
Examples

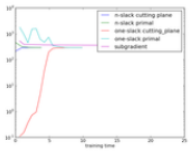

Plotting the objective and constraint caching in 1-slack SSVM

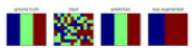

Efficient exact learning of 1-slack SSVMs

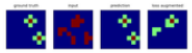

SVM as CRF


Semantic Image Segmentation on Pascal VOC

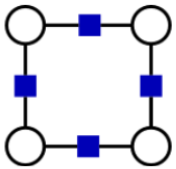

Latent Dynamics CRF


SVM objective values

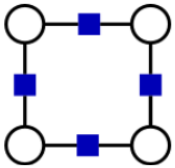

Learning directed interactions on a 2d grid


Learning interactions on a 2d grid

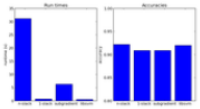

OCR Letter sequence recognition

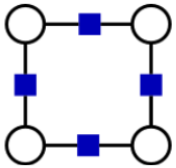

Crammer-Singer Multi-Class SVM


Latent SVM for odd vs. even digit classification


Mult-label classification


Latent Variable Hierarchical CRF


Binary SVM as SSVM


Comparing PyStruct and SVM-Struct

Back to top

http://pystruct.github.io

Summary of Contributions

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.
- Well tested, efficient software implementation for learning structured prediction. Already used by other researchers.

Future Directions

Future Directions

- More complex models for semantic segmentation.
- Integrate over time in NYU dataset.

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.

Additional Slides Exact Learning

n-Slack Cutting Plane Training of Structural SVMs

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

Ensure: parameters θ, slack $(\xi_1, \ldots, \xi_k)$

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

2: **repeat**

3: **for** i=1, . . . , k **do**

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

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:

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

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

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

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

1-Slack Cutting Plane Training of Structural SVMs

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

Ensure: parameters θ, slack ξ

$\mathcal{W} \leftarrow \emptyset$

repeat

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

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

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

for i=1, . . . , k **do**

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

$\mathcal{W} \leftarrow \mathcal{W} \cup \{(\hat{y}^i, \ldots, \hat{y}^i)\}$

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

until $\xi' - \xi < \epsilon$

Block-Coordinate Frank-Wolfe Algorithm

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

Ensure: parameters θ

$\theta_0, \theta_0^i, \bar{\theta}_0 \leftarrow \mathbf{0}, \quad \ell_0, \ell_0^i, t \leftarrow 0$

repeat

$t \leftarrow t + 1$

Pick i uniformly at random from $\{1, \ldots, k\}$.

Perform loss-augmented prediction on sample i:

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

Compute parameter and loss updates based on sample i:

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

Compute optimum step size η:

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

Update per-sample parameters and loss estimate:

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

Update global parameters and loss estimate:

$$\theta_{t+1} \leftarrow \theta_{t+1} + \theta_t^i - \theta_{t+1}^i$$
$$\ell_{t+1} \leftarrow \ell_{t+1} + \ell_t^i - \ell_{t+1}^i$$

Compute the weighted running average:

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

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

where θ_s and ℓ_s are recomputed over the whole dataset.