

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



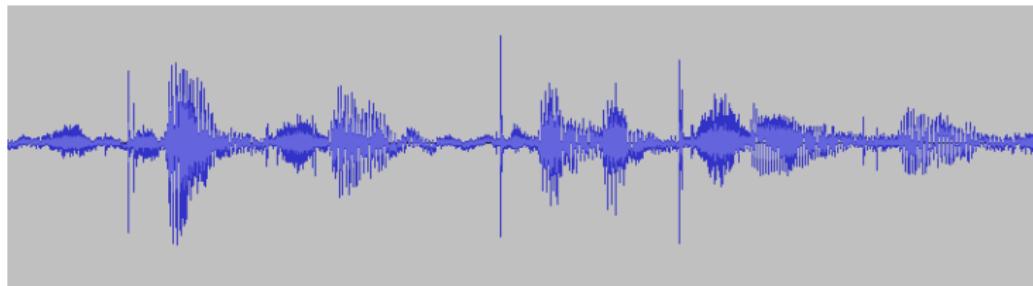
Stroke cat.

Stroke cat.

Stroke cat.

Open trash can.

Put cat in trash can.



Sequence Tagging



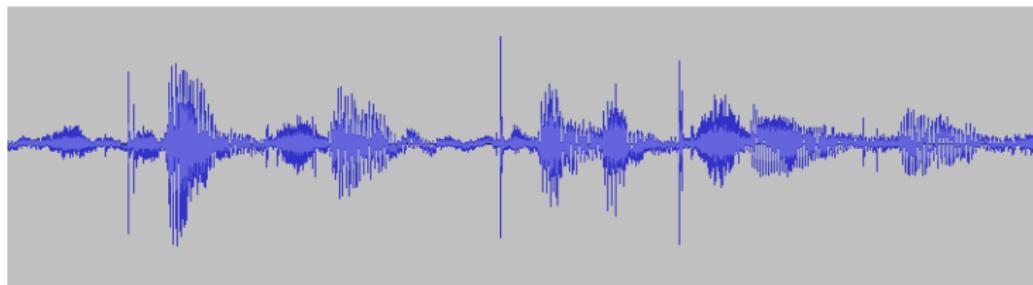
Stroke cat.

Stroke cat.

Stroke cat.

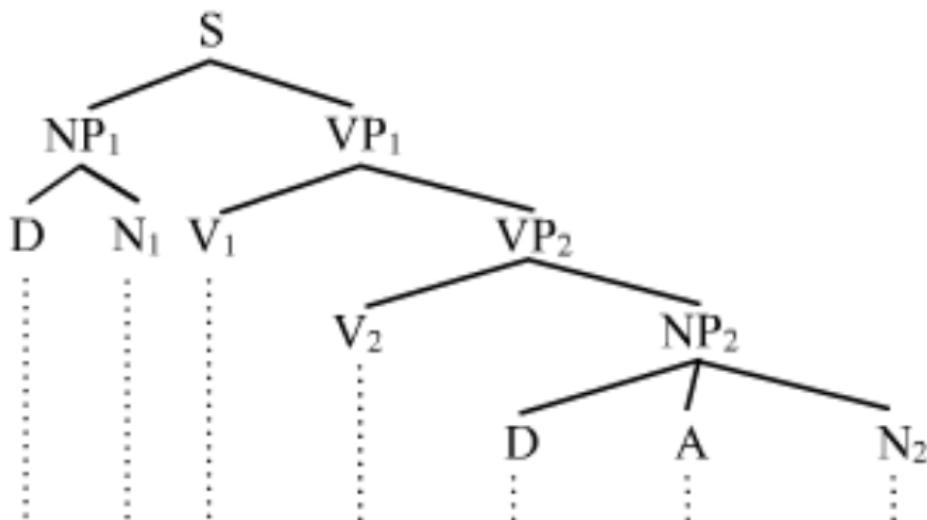
Open trash can.

Put cat in trash can.



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

Predicting Parse Trees



This tree is illustrating the constituency relation.

Constituency relation (PSG)

Structured Prediction for Semantic Segmentation

Semantic Segmentation as Structured Prediction



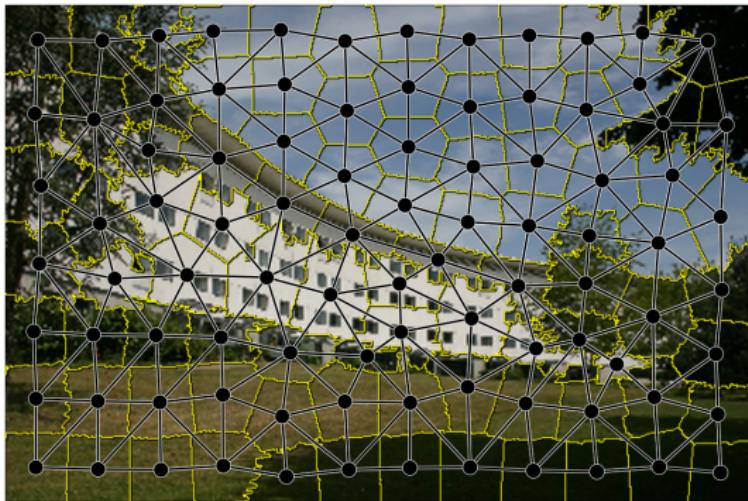
Semantic Segmentation as Structured Prediction



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

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

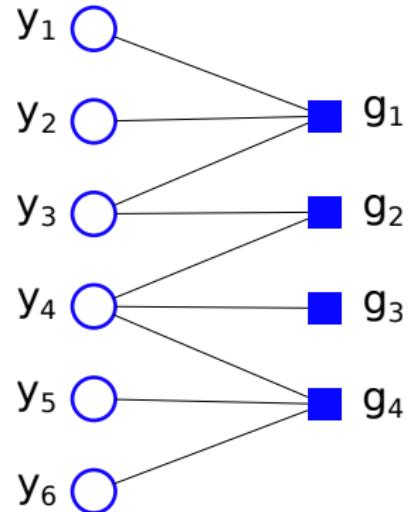
Predicting discrete vectors

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

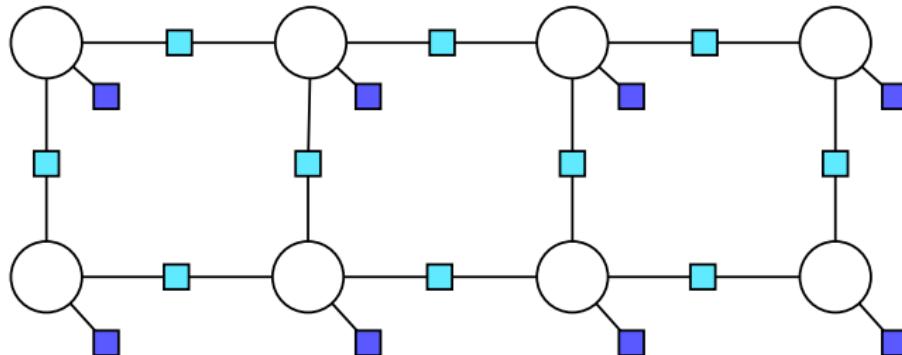
$$\begin{aligned} f(x, w) &= \arg \max_{y \in \mathcal{Y}} w^T \psi(x, y) \\ &= \arg \max_{y_1, y_2, \dots, y_{n_i}} w^T \psi(x, y) \end{aligned}$$

Factor Graphs

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

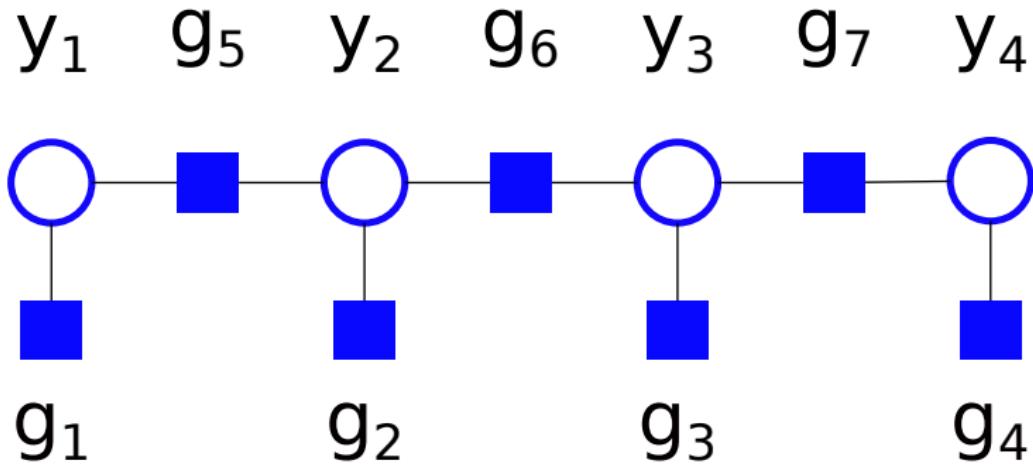


Pairwise Models



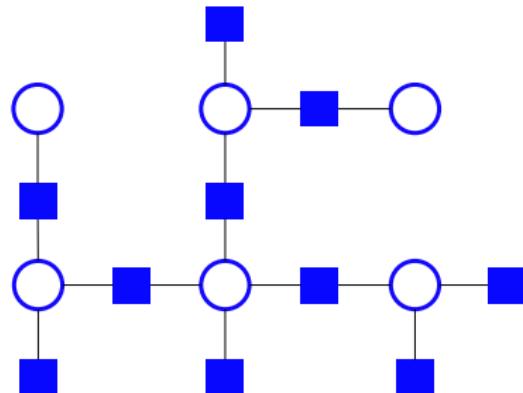
$$w^T \psi(x, y) = \sum_{(i,j) \in E} w_{i,j} \psi_{i,j}(x, y_i, y_j) + \sum_{i \in I} w_i \psi_i(x, y_i)$$

Factor Graph for HMM

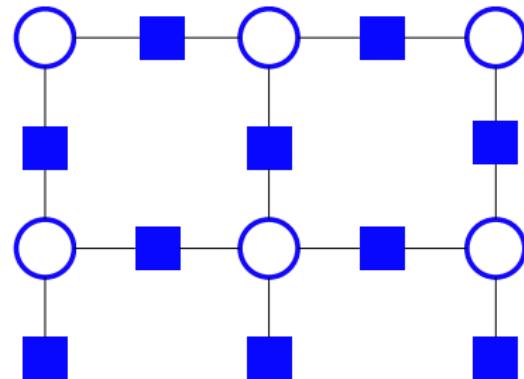


Inference

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

$$\begin{aligned} & \max_w \sum_i \log(p(y^i|x^i, w)) \\ &= \max_w \sum_i w^T \psi(x^i, y^i) - \log(Z) \end{aligned}$$

Max-Margin Learning

Learn prediction function of the form

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

Max-Margin Learning

Learn prediction function of the form

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

Objective

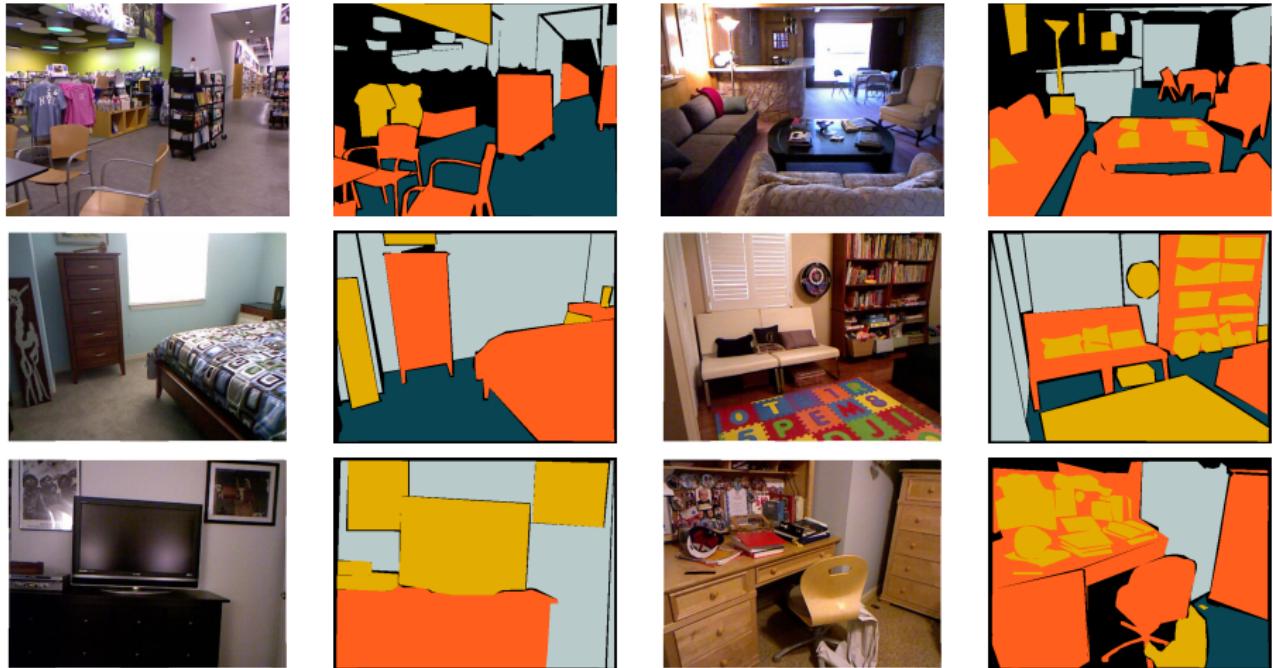
$$\min_w \frac{1}{2} \|w\|^2 + C \sum_i \ell(x^i, y^i, w)$$

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

Joachims, Finley, and Yu 2009

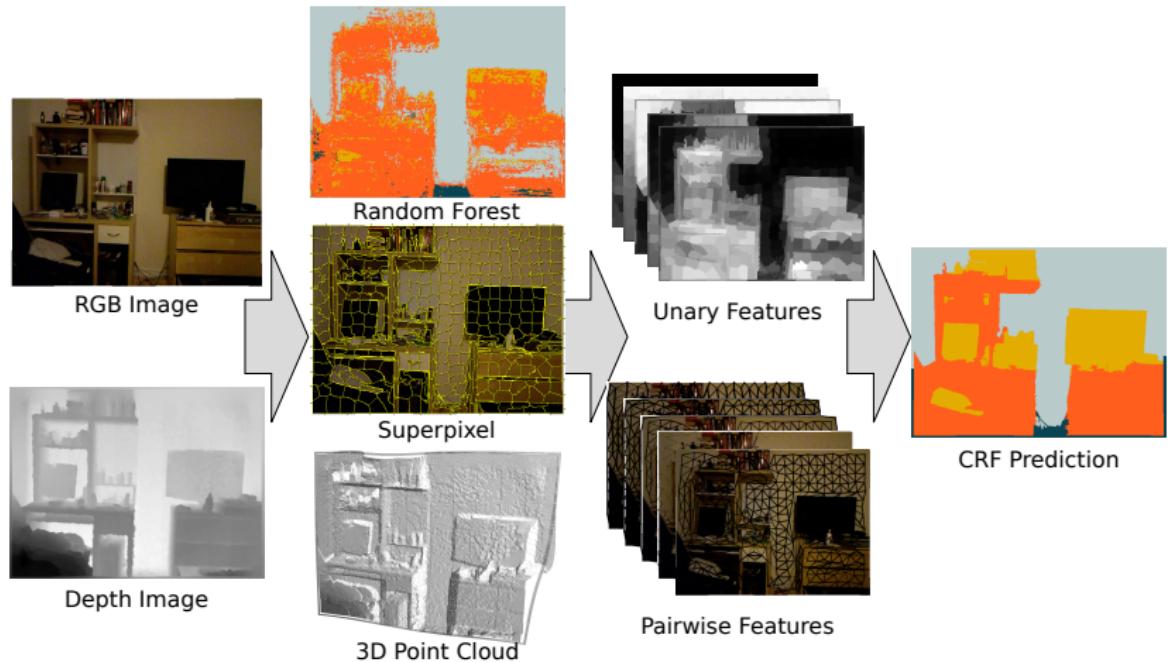
Learning Depth-Sensitive Conditional Random Fields

Dataset: NYUv2



795 training images, 654 test images.

Overview



Pairwise Features

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



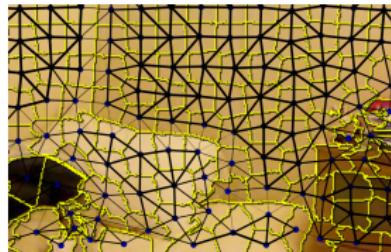
$$f_{\text{depth}}(x_i, x_j) = (d_i - d_j)/Z$$



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



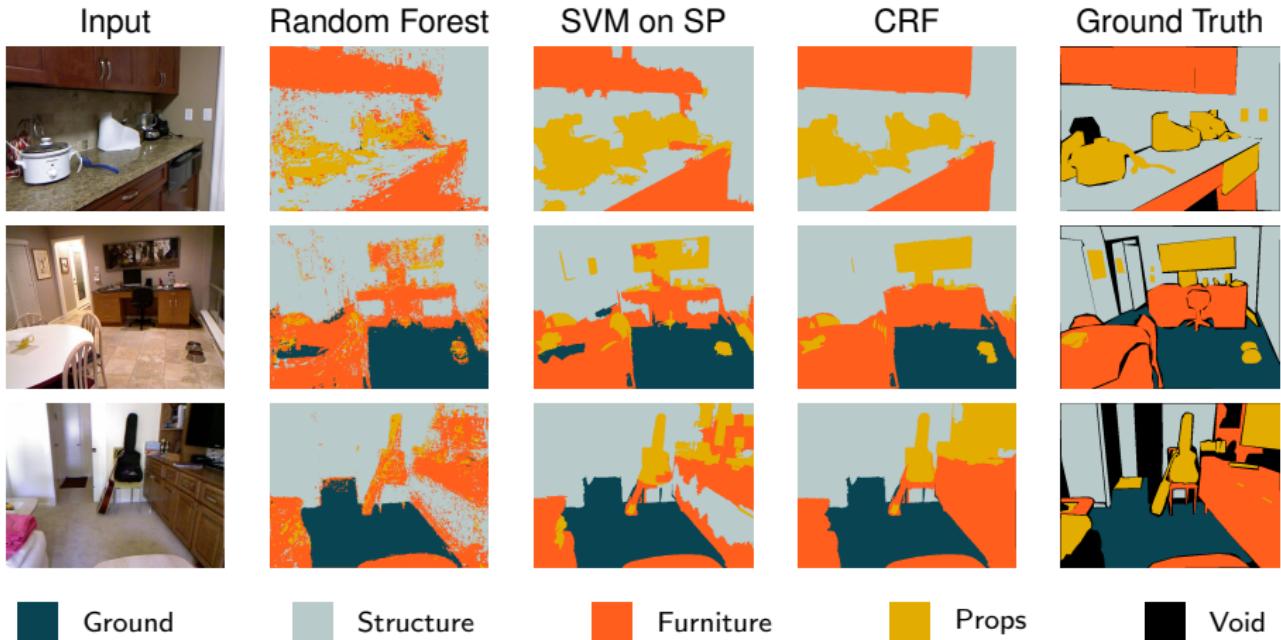
$$f_{\text{normals}}(x_i, x_j) = 1 - \frac{1}{\pi} \langle \mathbf{n}_{x_i}, \mathbf{n}_{x_j} \rangle$$



Learning and Optimization

- 1-slack SSVM
- Inference using fusion moves and AD³.
- Exact learning in loopy model (Müller and Behnke 2014).

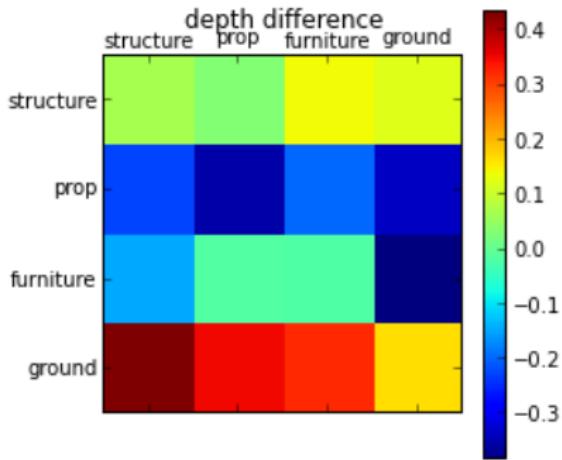
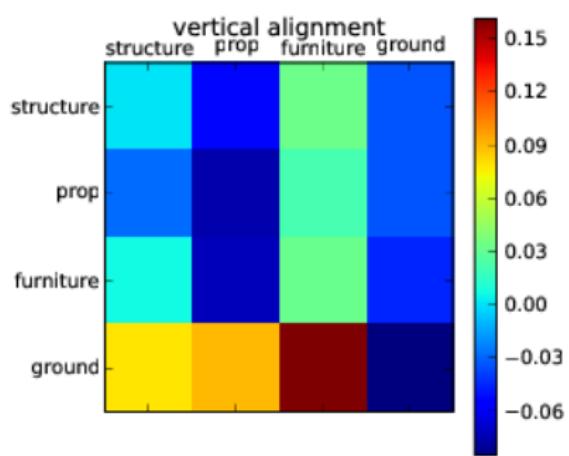
Qualitative Results



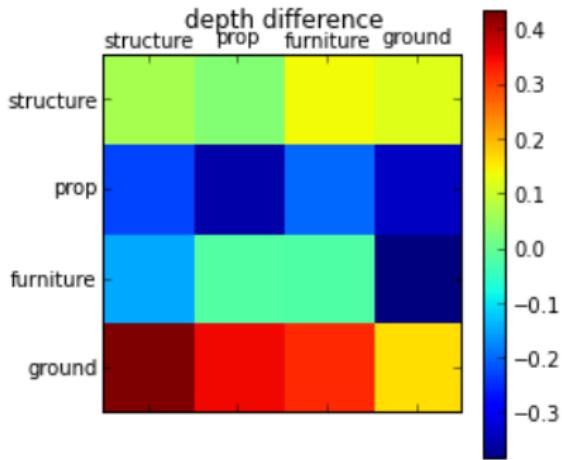
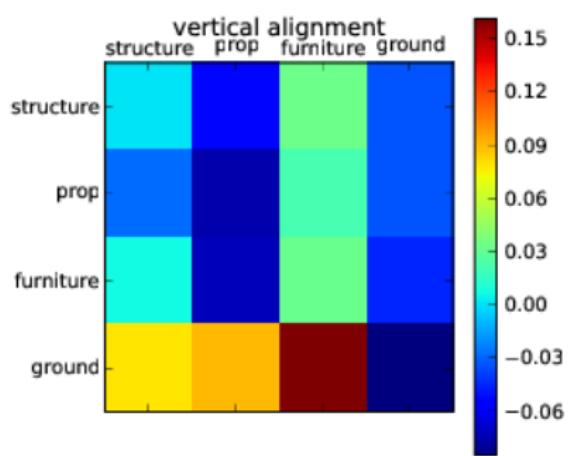
Results

	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
Silberman et al. (2012)	68	59	70	42	59.6	58.6
Couprise et al. (2013)	87.3	86.1	45.3	35.5	63.5	64.5
Stückler et al. (2013) [†]	95.6	83.0	75.1	14.2	67.0	70.9

Learned Potentials



Learned Potentials



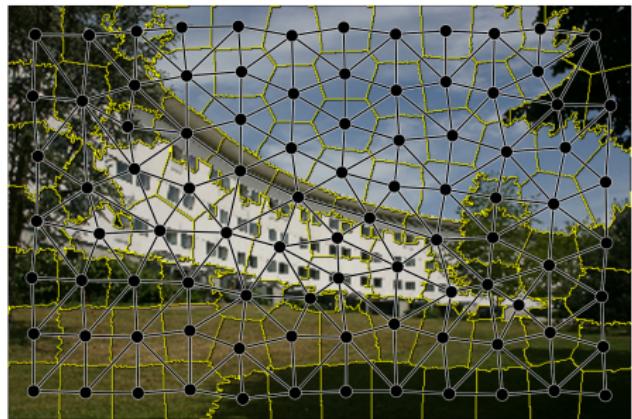
- Ground is at bottom.
- Walls are behind other objects.
- Ground is often in front of other objects.
- Furniture is in front of the walls.

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

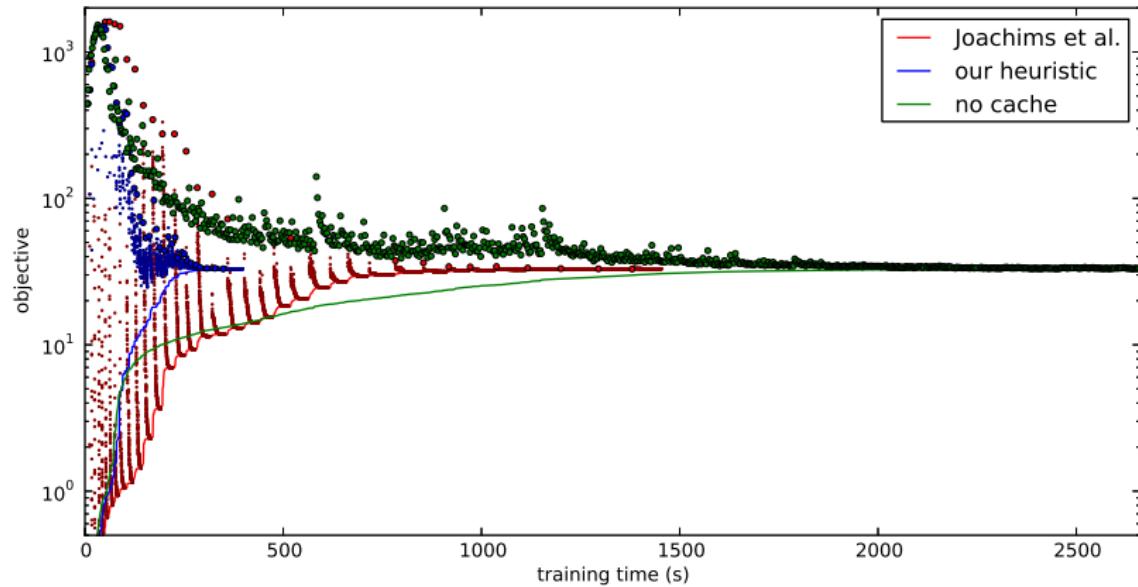
$$\ell(x^i, y^i, w) = [\max_{y \in \mathcal{Y}} \Delta(y^i, y) + w^T \psi(x^i, y) - w^T \psi(x^i, y^i)]_+ \quad (1)$$

- 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

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

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

Caching Comparison



Combining Inference Algorithms

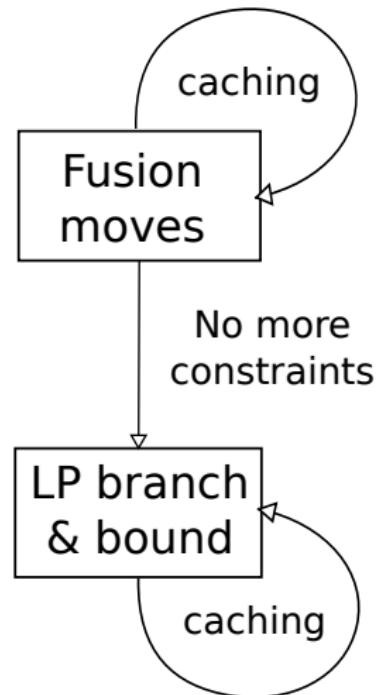
- Use fast inference in the beginning, use good inference at the end.

Combining Inference Algorithms

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

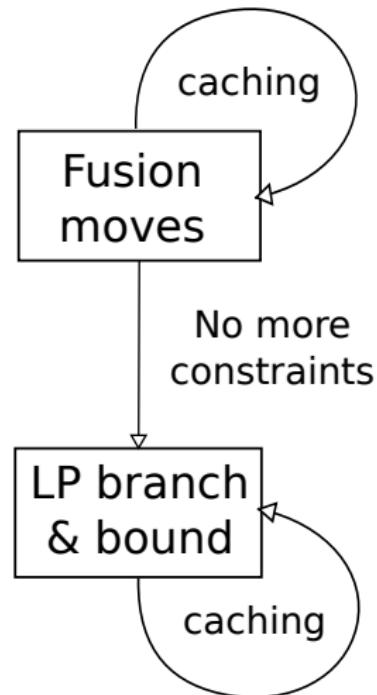
Combining Inference Algorithms

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

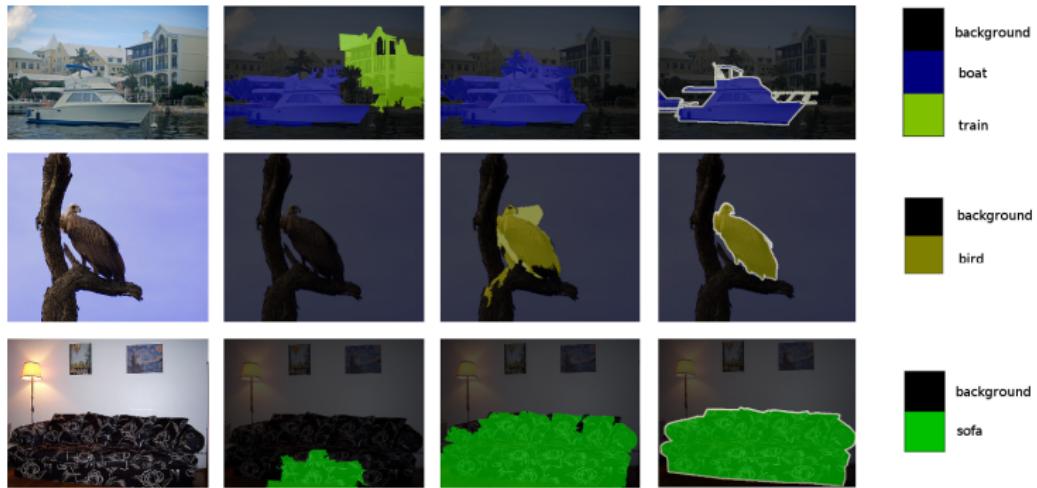


Combining Inference Algorithms

- 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
Dann et al. (2012)	27.4
Krähenbühl and Koltun (2012)	30.2
Krähenbühl and Koltun (2013)	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
Ladicky et al. (2009)	75.8	85.0
Gonfaus et al. (2010)	77	75
Lucchi, Li, and Fua (2013)	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

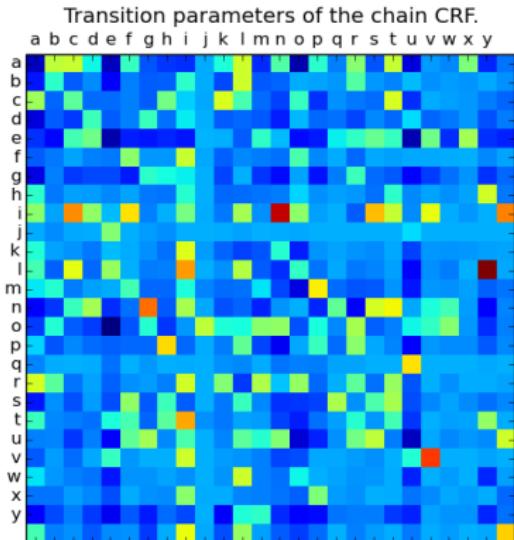
abc = "abcdefghijklmnopqrstuvwxyz"

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]

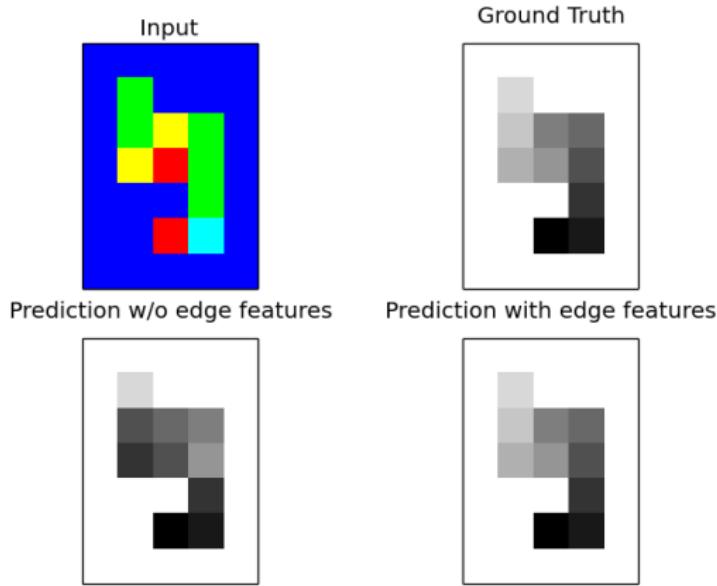
# Train linear SVM
svm = LinearSVC(dual=False, C=.1)
# flatten input
svm.fit(np.vstack(X_train), np.hstack(y_train))

# 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



Example Snake



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.

Thank you

Additional Slides Exact Learning

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

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

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

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 :

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

8: Update per-sample parameters and loss estimate:

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

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

9: Update global parameters and loss estimate:

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

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

10: Compute the weighted running average:

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

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

where θ_s and ℓ_s are recomputed over the whole dataset.

References

-  Camille Couprie et al. "Indoor Semantic Segmentation using Depth Information". In: *International Conference on Learning Representations*. 2013.
-  Christoph Dann et al. "Pottics—The Potts Topic Model for Semantic Image Segmentation". In: *German Conference on Pattern Recognition (DAGM)*. 2012.
-  Josep M Gonfaus et al. "Harmony potentials for joint classification and segmentation". In: *Computer Vision and Pattern Recognition*. 2010.
-  Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. "Cutting-plane training of structural SVMs". In: *Machine Learning* 77.1 (2009).
-  Philipp Krähenbühl and Vladlen Koltun. "Efficient inference in fully connected CRFs with Gaussian edge potentials". In: 2012.
-  Philipp Krähenbühl and Vladlen Koltun. "Parameter Learning and Convergent Inference for Dense Random Fields". In: *International Conference on Machine Learning*. 2010.