Introduction to Structured Prediction (with Python)

Andreas C. Müller

Columbia University

July 3, 2018

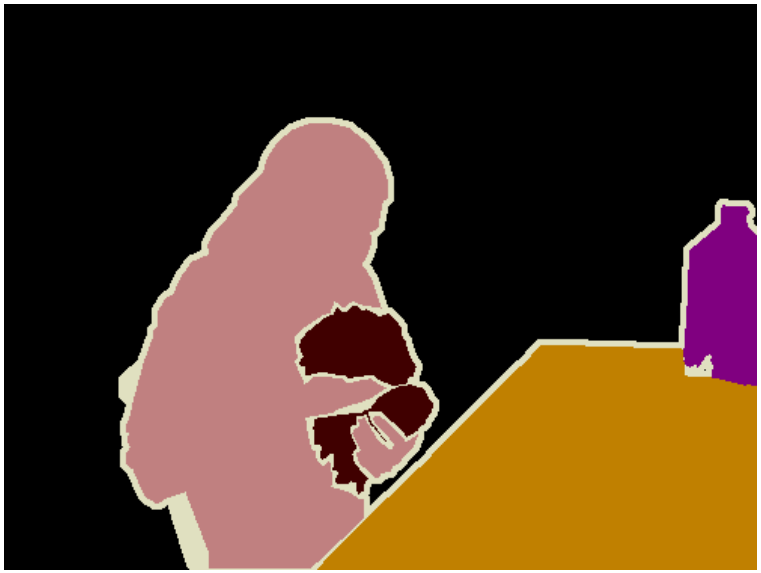
1. Inference Algorithms

2. Learning Algorithms
   - Optimizing Structural SVMs

3. Structured Prediction with PyStruct

Semantic Segmentation

Semantic Segmentation



Multi-Label Classification

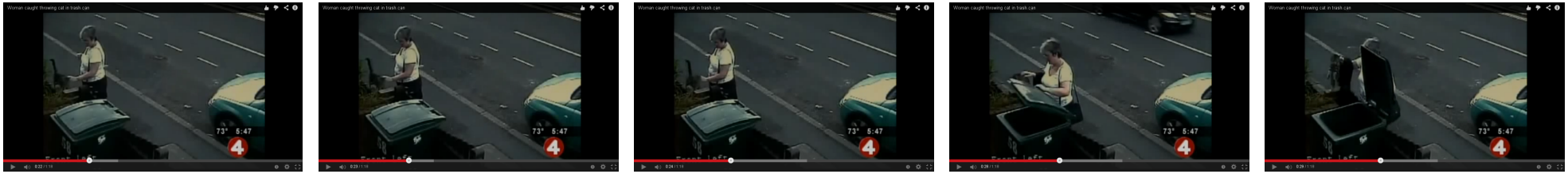
	Politics	Sports	Finance	Domestic	Religion
News Story1	1	0	0	1	1
News Story2	0	1	0	1	0
News Story3	0	0	1	0	0

Multi-Label Classification

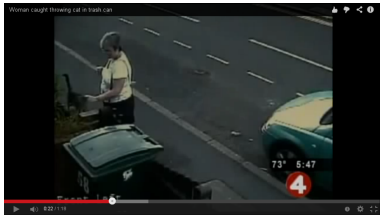
	Politics	Sports	Finance	Domestic	Religion
News Story1	1	0	0	1	1
News Story2	0	1	0	1	0
News Story3	0	0	1	0	0

	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

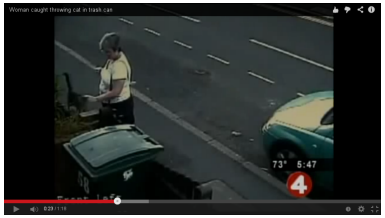
Sequence Tagging



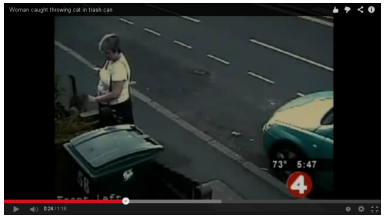
Sequence Tagging



Stroke cat.



Stroke cat.

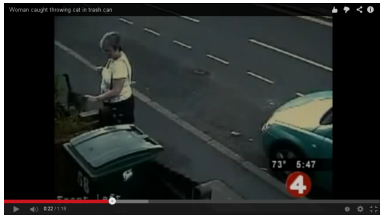


Stroke cat.

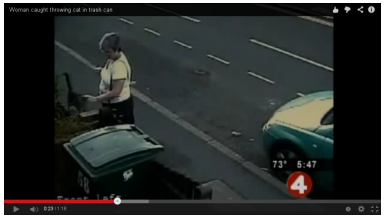
Open trash can.

Put cat in trash can.

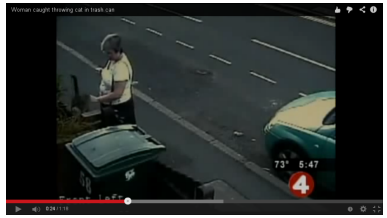
Sequence Tagging



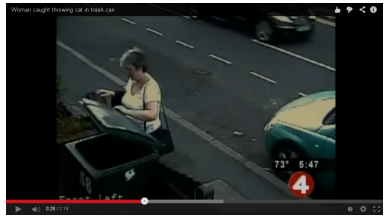
Stroke cat.



Stroke cat.

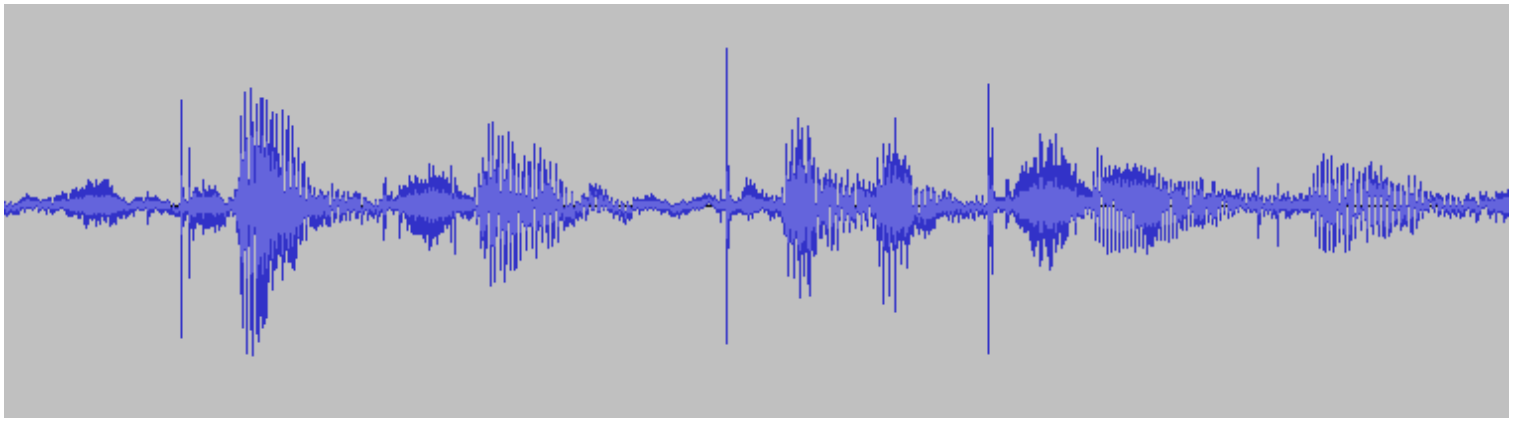


Stroke cat.

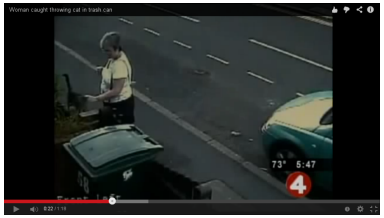


Open trash can.

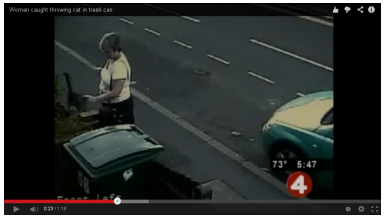
Put cat in trash can.



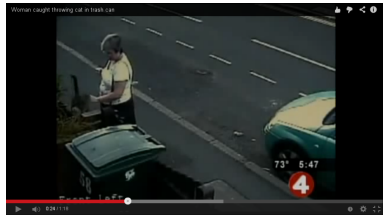
Sequence Tagging



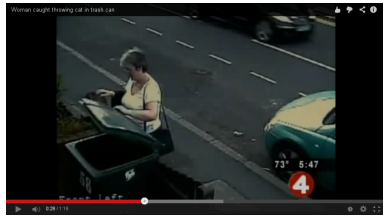
Stroke cat.



Stroke cat.

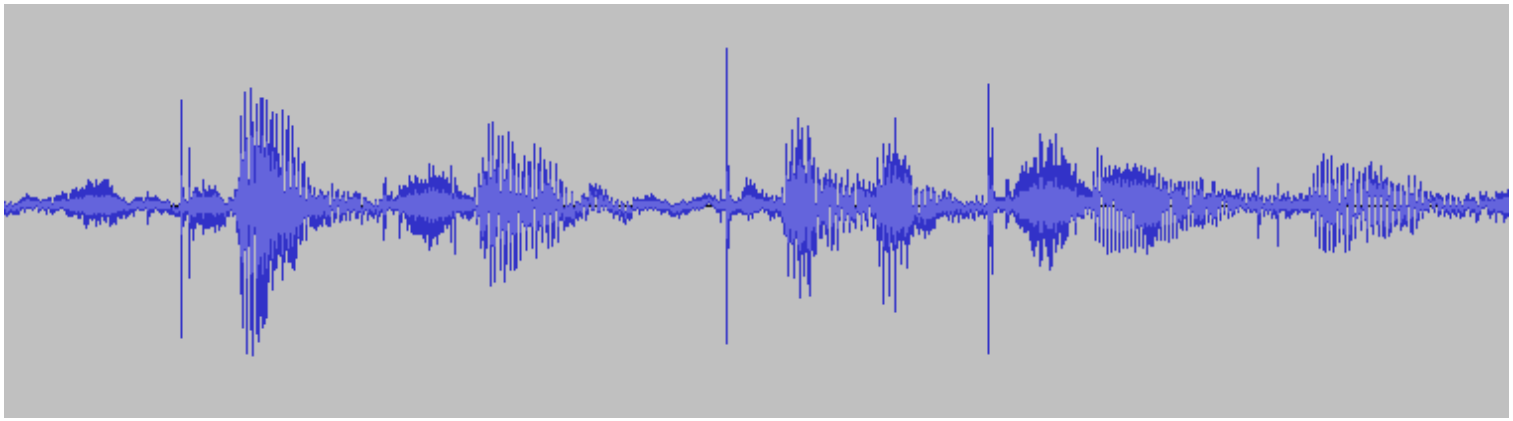


Stroke cat.



Open trash can.

Put cat in trash can.



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

Predicting 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

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

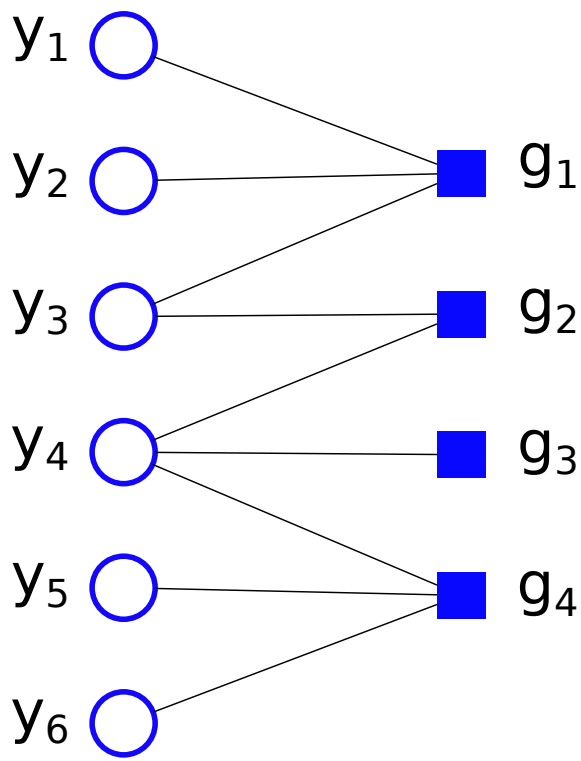
Predicting discrete vectors

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

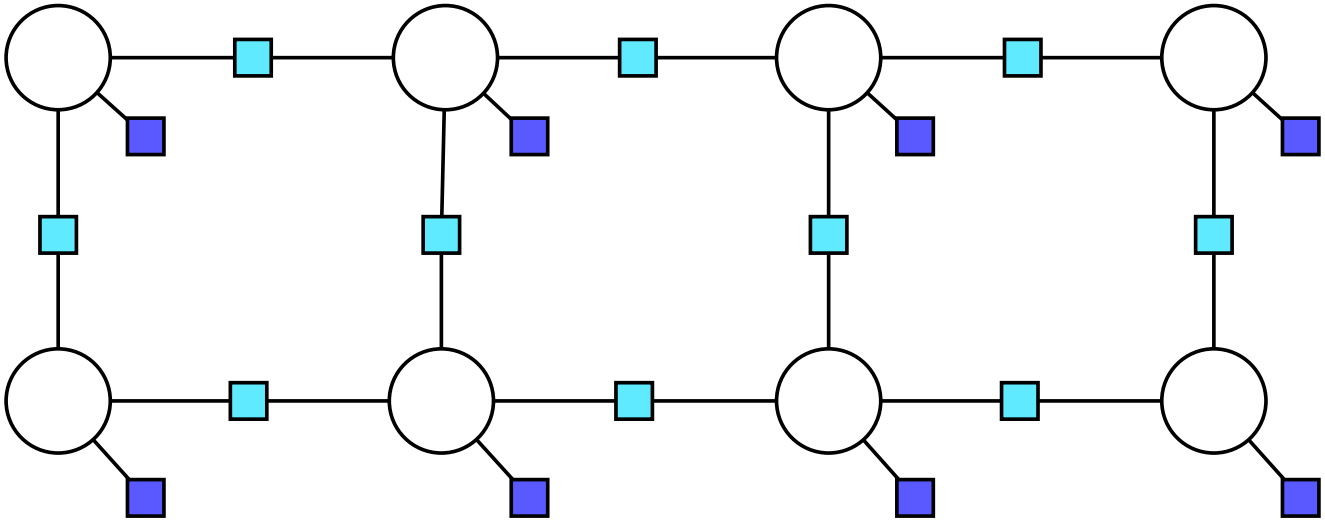
$$\begin{aligned} f(x, w) &= \arg\max_{y \in \mathcal{Y}} w^T \psi(x, y) \\ &= \arg\max_{y_1, y_2, \ldots, y_{n_i}} w^T \psi(x, y) \end{aligned}$$

Factor Graphs

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

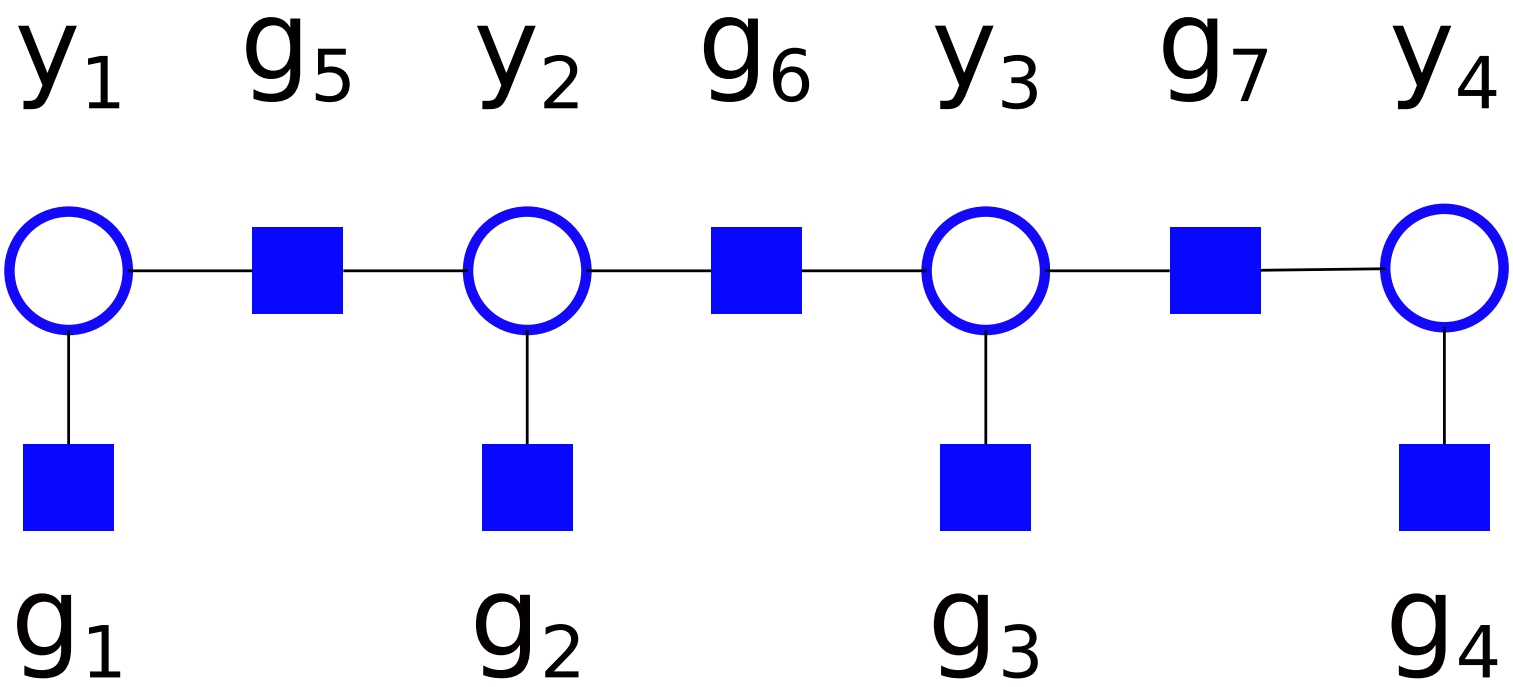


Pairwise Models



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

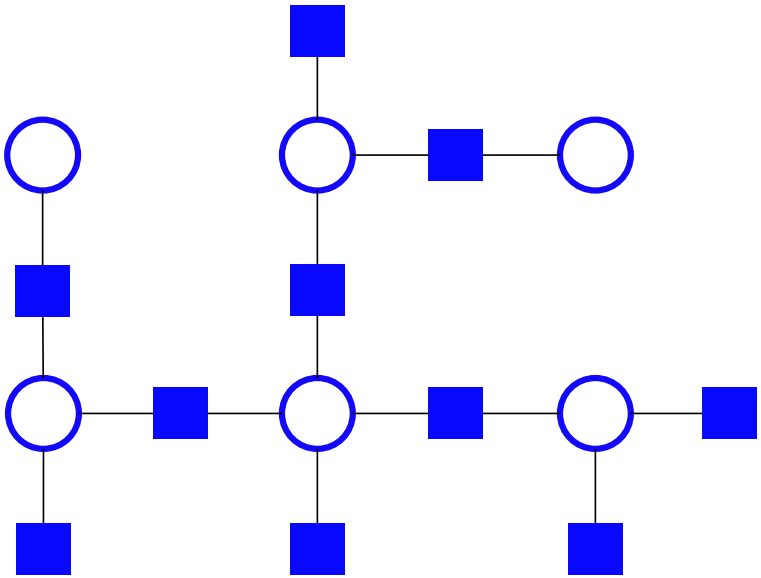
Factor Graph for HMM



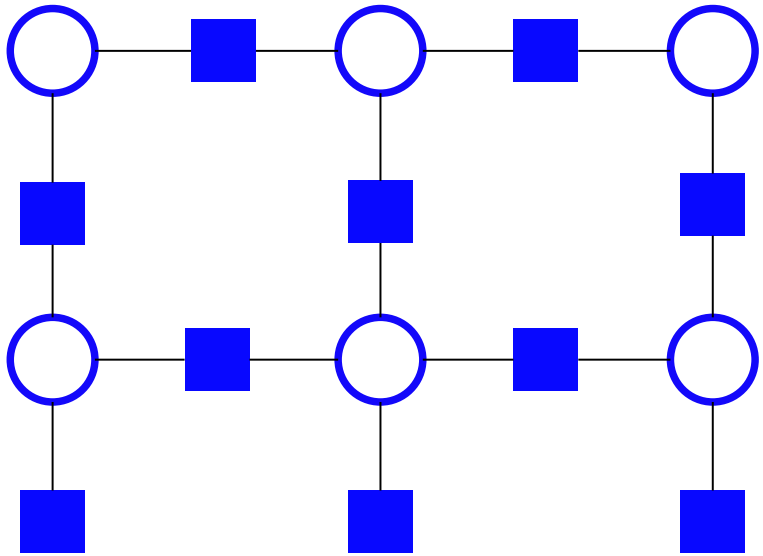
Factor Graphs Multi-Label Classification

Inference

Easy



Tricky



Inference Algorithms

Brute Force

Sum-Product (Viterbi)

Sampling

Relaxations

Move-making

Learning Algorithms

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

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

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

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

Compare to Binary SVM

Binary

$$\min_w \frac{1}{2}\|w\|^2 + C\sum_i \max(0, 1 - y^i w^T x^i)$$

Compare to Binary SVM

Binary

$$\min_w \frac{1}{2}\|w\|^2 + C\sum_i \max(0, 1 - y^i w^T x^i)$$

Crammer-Singer Multi-Class

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

$$\ell(x^i, y^i, w) = [1 + \max_{\hat{y}!=y_i} w_{\hat{y}}^T x^i - w_{y^i}^T x^i]_+.$$

Compare to Binary SVM

Binary

$$\min_w \frac{1}{2}\|w\|^2 + C\sum_i \max(0, 1 - y^i w^T x^i)$$

Crammer-Singer Multi-Class

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

$$\ell(x^i, y^i, w) = [1 + \max_{\hat{y}!=y_i} w_{\hat{y}}^T x^i - w_{y^i}^T x^i]_+.$$

Structured

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

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

Subgradient learning

Require: training samples $\{(x^1, y^1), \ldots, (x^k, y^k)\}$, regularization parameter C, learning rates η_t.

Ensure: parameters θ

$w \leftarrow 0$

repeat

  for $i = 1, \ldots, k$ **do** $\hat{y} \leftarrow \operatorname{argmax}_{\hat{y} \in \mathcal{Y}} \delta(y^i, \hat{y}) - \theta^T [\Phi(x^i, y^i) - \Phi(x^i, \hat{y})]$

    $w \leftarrow (1 - \eta_t) w - \eta_t C [\Phi(x^i, y^i) - \Phi(x^i, \hat{y})]$

until stopping tolerance reached.

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 $(\xi_1, \ldots, \xi_k)$

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

repeat

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

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

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

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

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

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

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

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**

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

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

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

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

Structured Prediction with PyStruct

Simple structured prediction

Estimator = Learner + Model + Inference

Simple structured prediction

Estimator = Learner + Model + Inference

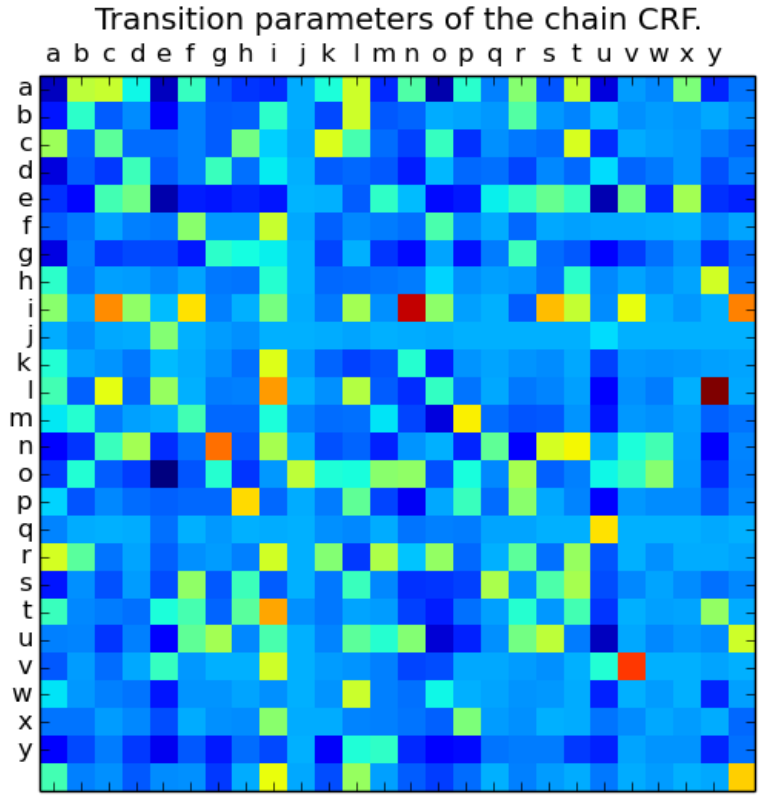
- Learner: SubgradientSSVM, StructuredPerceptron, OneSlackSSVM, LatentSSVM
- Model: BinaryClf, MultiLabelClf, ChainCRF, GraphCRF, EdgeFeatureGraphCRF
- Inference: Linear Programming, QPBO (PyQPBO), Dual Decomposition (AD3), Message Passing, Everything (OpenGM)

Example OCR

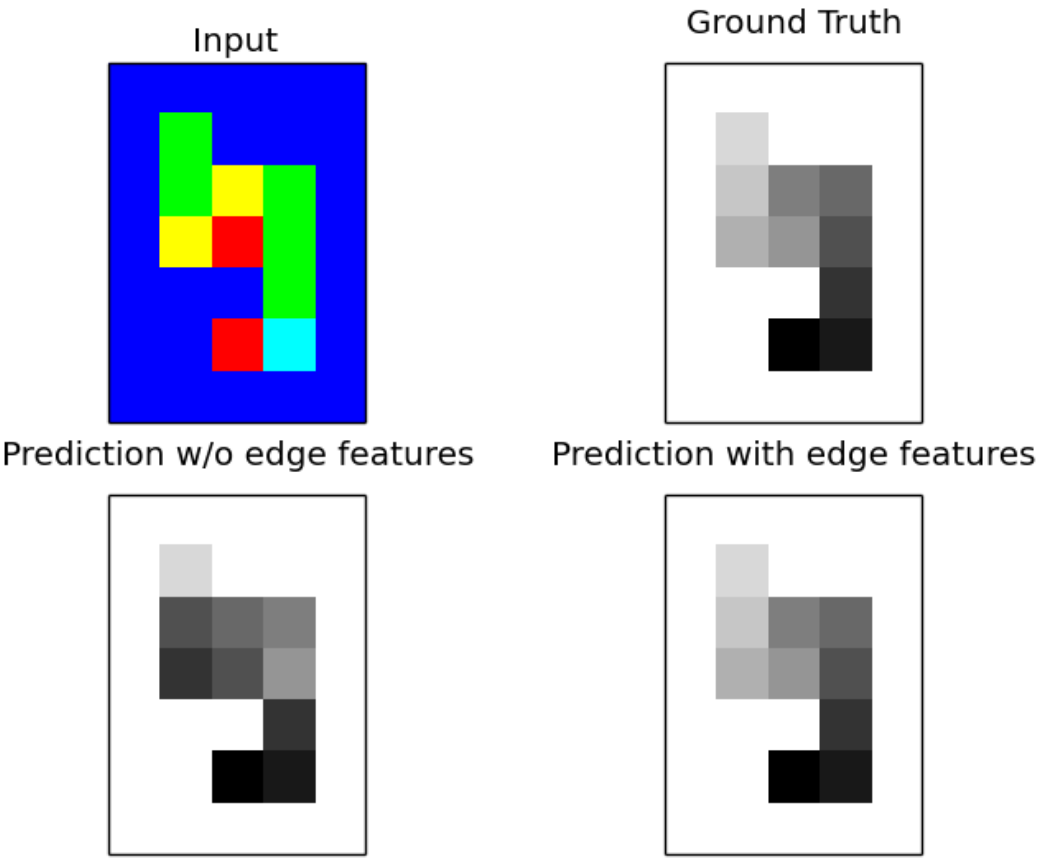
```
from pystruct.datasets import load_letters
from pystruct.models import ChainCRF
from pystruct.learners import OneSlackSSVM

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

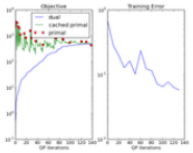

a f e t e r i a
a a f t a a f t m a r r i i a a
m b r a c e s
m m b b r r a a c c e e s s
p o i l f n g
p p a a i i c c i i m n d g
a f e t e r i a
a a f f o o d d o o l l z i n n


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

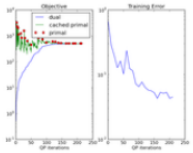
Example Snake



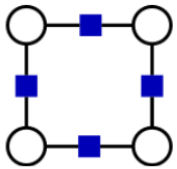
Examples



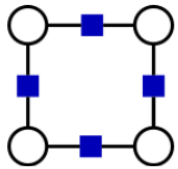
Plotting the objective and constraint caching in 1-slack SSVM



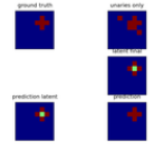
Efficient exact learning of 1-slack SSVMs



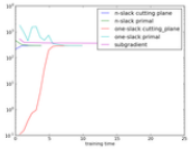
SVM as CRF



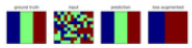
Semantic Image Segmentation on Pascal VOC



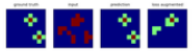
Latent Dynamics CRF



SVM objective values

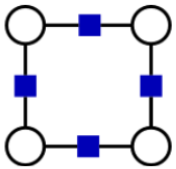


Learning directed interactions on a 2d grid



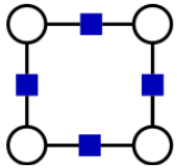
Learning interactions on a 2d grid

OCR Letter sequence recognition



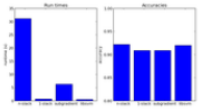
Crammer-Singer Multi-Class SVM

Latent SVM for odd vs. even digit classification

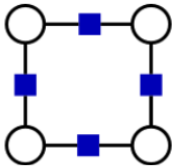


Mult-label classification

Latent Variable Hierarchical CRF



Binary SVM as SSVM



Comparing PyStruct and SVM-Struct

Back to top

http://pystruct.github.io

Thank you

References


Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. “Cutting-plane training of structural SVMs”. In: *Machine Learning* 77.1 (2009).