

Introduction to Structured Prediction (with Python)

Andreas C. Müller

Columbia University

July 2, 2018

- 1 Inference Algorithms
- 2 Learning Algorithms
- 3 Structured Prediction with PyStruct
- 4 Semantic Segmentation and Edge Features
- 5 Latent Variable Models

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



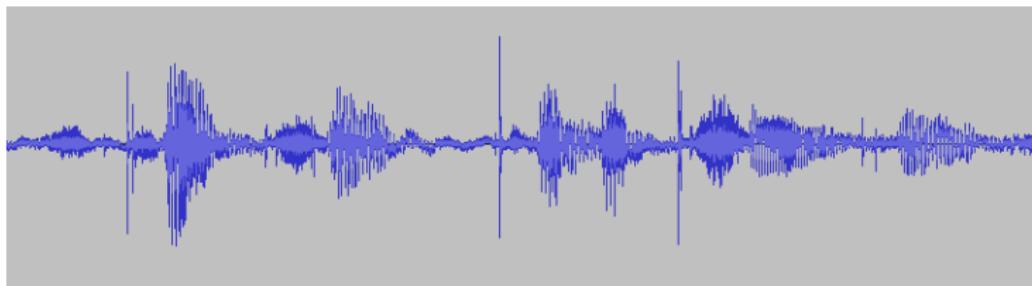
Stroke cat.

Stroke cat.

Stroke cat.

Open trash can.

Put cat in trash can.



Sequence Tagging



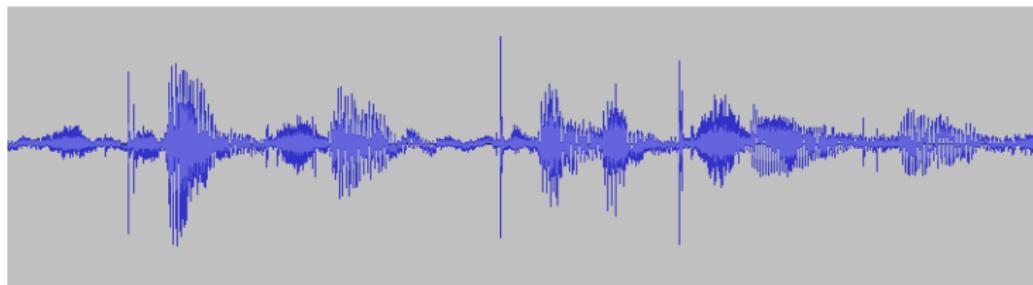
Stroke cat.

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

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

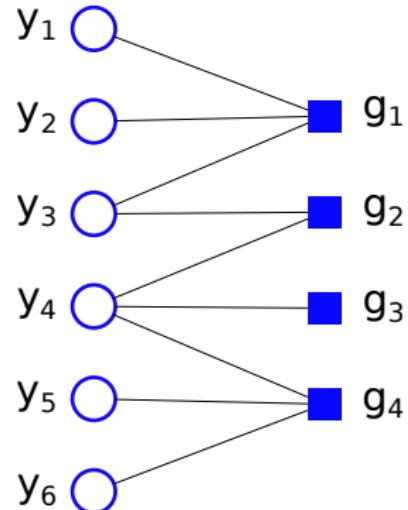
Predicting discrete vectors

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

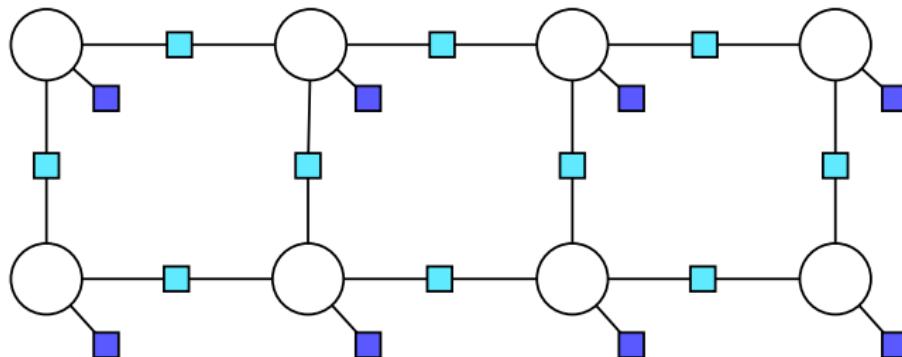
$$\begin{aligned} f(x, w) &= \arg \max_{y \in \mathcal{Y}} w^T \psi(x, y) \\ &= \arg \max_{y_1, y_2, \dots, y_{n_i}} w^T \psi(x, y) \end{aligned}$$

Factor Graphs

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

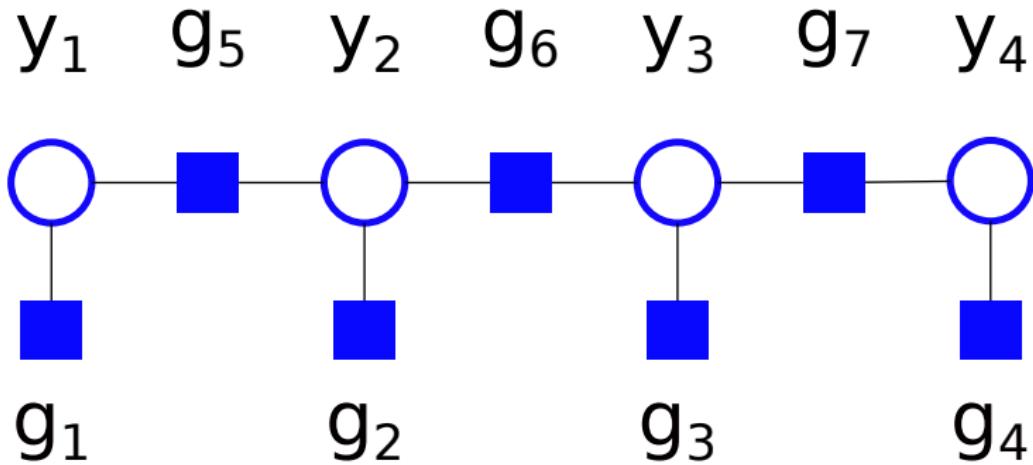


Pairwise Models



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

Factor Graph for HMM

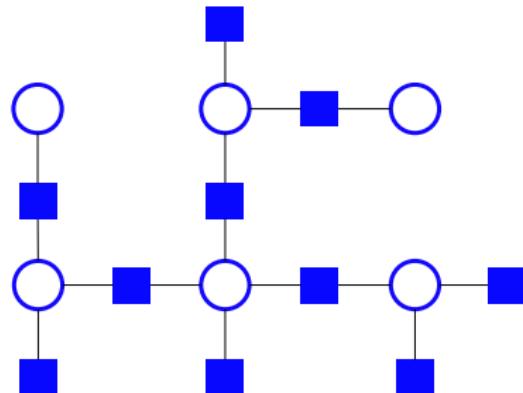


Factor Graphs Mulit-Label Classification

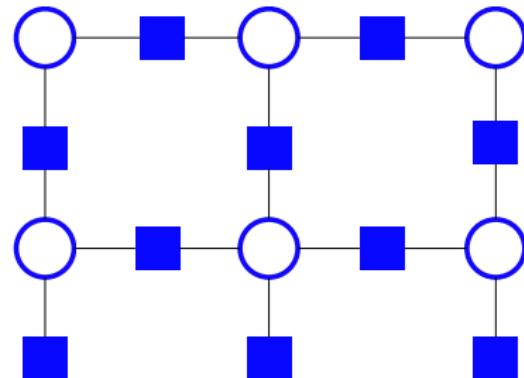
FIXME full, chow liu

Inference

Easy



Tricky



Inference Algorithms

Brute Force

Sum-Product (Viterbi)

Sampling

Relaxations

Move-making

Learning Algorithms

Probabilistic Learning

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

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

FIXME relate to LogisticRegression

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

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

FIXME relate to LogisticRegression

Max-Margin Learning

Learn prediction function of the form

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

FIXME relate to SVM Advantage: no

partition function

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 FIXME relate to SVM Advantage: no partition function

Subgradient learning

N-Slack Quadratic Programming

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 Quadratic Programming

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** $\xi' - \xi < \epsilon$

Frank Wolfe

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.

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

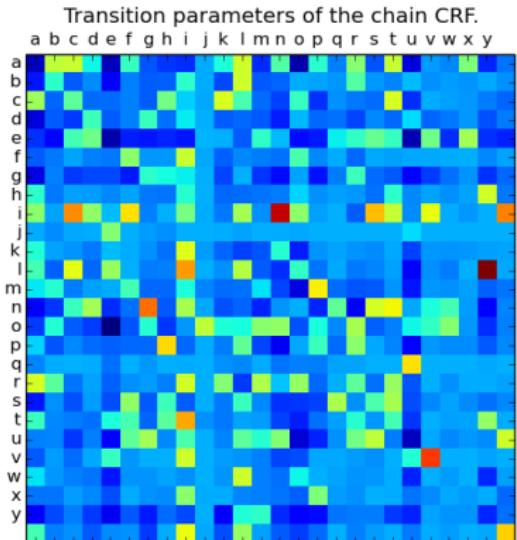
abc = "abcdefghijklmnopqrstuvwxyz"

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]

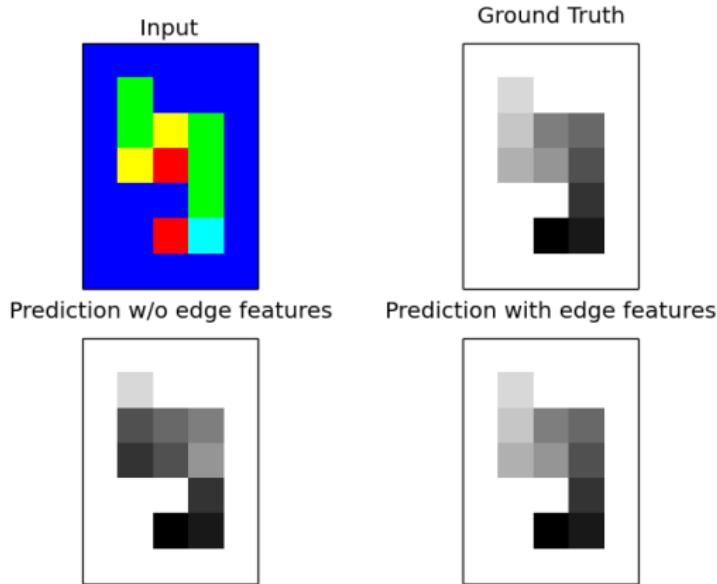
# Train linear SVM
svm = LinearSVC(dual=False, C=.1)
# flatten input
svm.fit(np.vstack(X_train), np.hstack(y_train))

# 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)
```

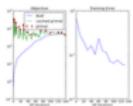
Example OCR



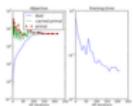
Example Snake



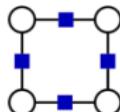
Examples



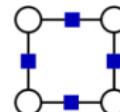
Plotting the objective and constraint caching in 1-slab SSVM



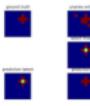
Efficient exact learning of 1-slab SSVMs



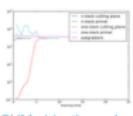
SVM as CRF



Semantic Image Segmentation on Pascal VOC



Latent Dynamics CRF



SVM objective values



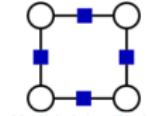
Learning directed interactions on a 2d grid



Learning interactions on a 2d grid



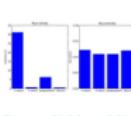
Latent SVM for odd vs. even digit classification



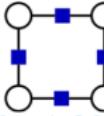
Mult-label classification



Latent Variable Hierarchical CRF



Binary SVM as SSVM



Comparing PyStruct and SVM-Struct

Multi-Class SVM

Multi-Label SVM (independent)

Multi-Label SVM (Chow Liu)

Multi-Label SVM (full)

Semantic Segmentation and Edge Features

Latent Variable Models

Thank you

References

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