10-418/10-618 Machine Learning for Structured Data

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Structured Perceptron

+

Structured SVM

Matt Gormley
Lecture 22
Nov. 21, 2022

Reminders

- **Homework 6: VAE + Structured SVM**
  - **Out: Wed, Nov 16**
  - **Due: Wed, Nov 24 at 11:59pm**

10-618 Mini-Project

What is it?

- The Mini-Project
  - Build a structured prediction system on a task from the MS CoCo dataset
  - Using existing code is fine, so long as the delta is clear
  - Starting from scratch is fine, and the difficulty of doing so will be taken into account
- Do not start early!
  - Intended to be a 15-20 hour effort, not more
  - A typical course project is 50-60 hours
- Release date: Mon, Nov 28

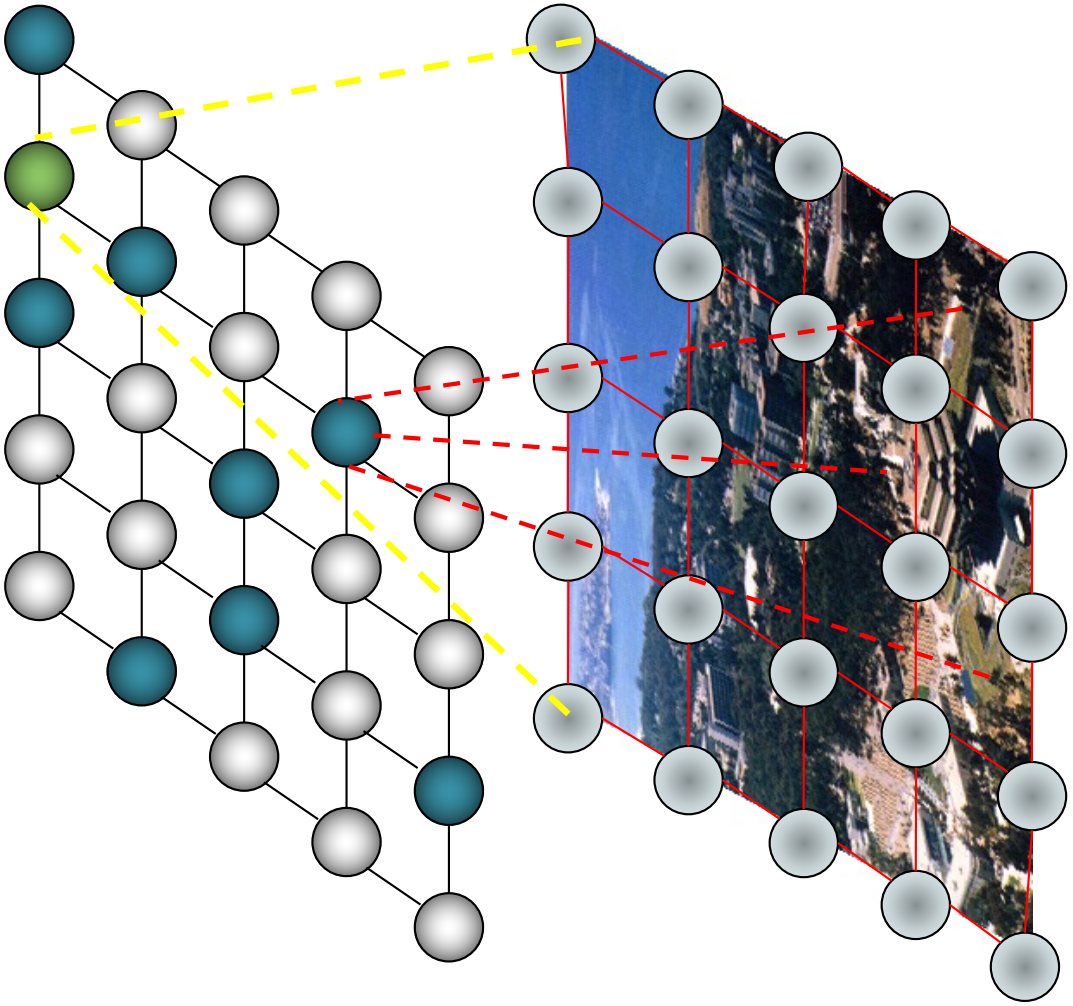
Milestones

- Team Formation
  - Due: Tue, Nov 29 (with wiggle room for stragglers)
- Proposal
  - Content: a few (very low stakes) paragraphs describing what you plan to do
  - Purpose: to get some early feedback from me
  - Due: Thu, Nov 30
- Executive Summary & Code Upload
  - Content: four pages
  - Purpose: to tell what you did
  - Due: Fri, Dec 9

Case #2: Multiclass Variables

MAP INFERENCE AS MATHEMATICAL PROGRAMMING

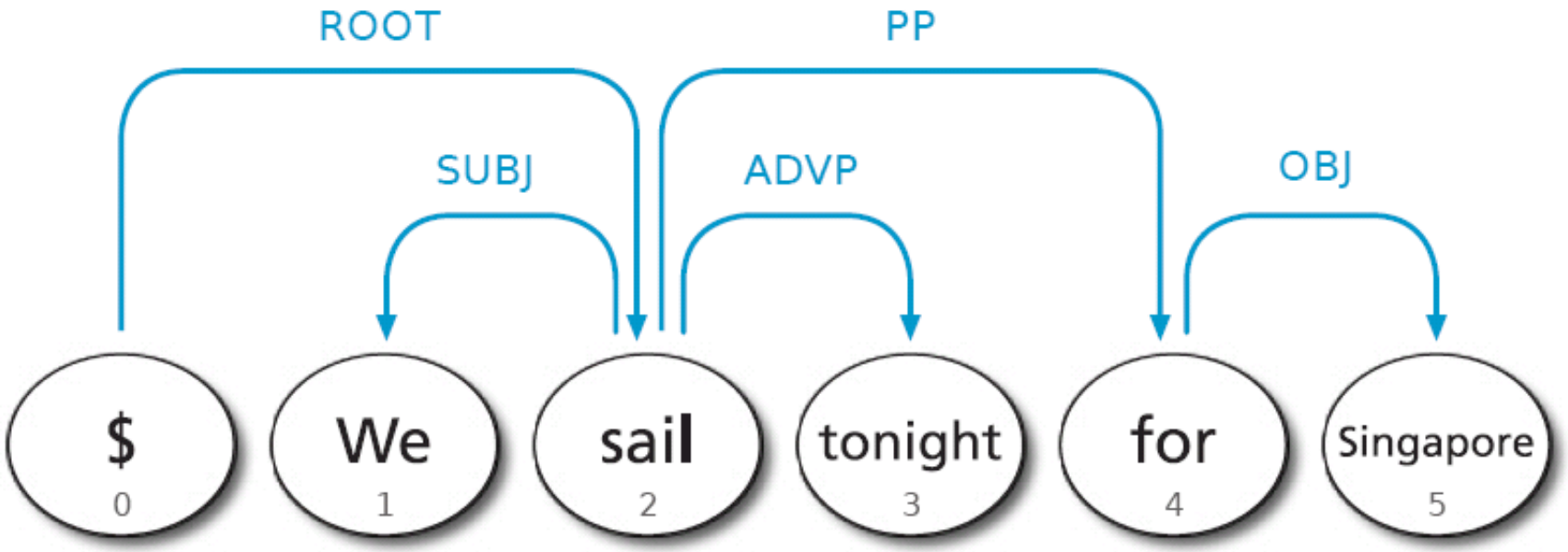
Image Segmentation



$$p_\theta(y \mid x) = \frac{1}{Z(\theta, x)} \exp\left\{\sum_c \theta_c f_c(x, y_c)\right\}$$

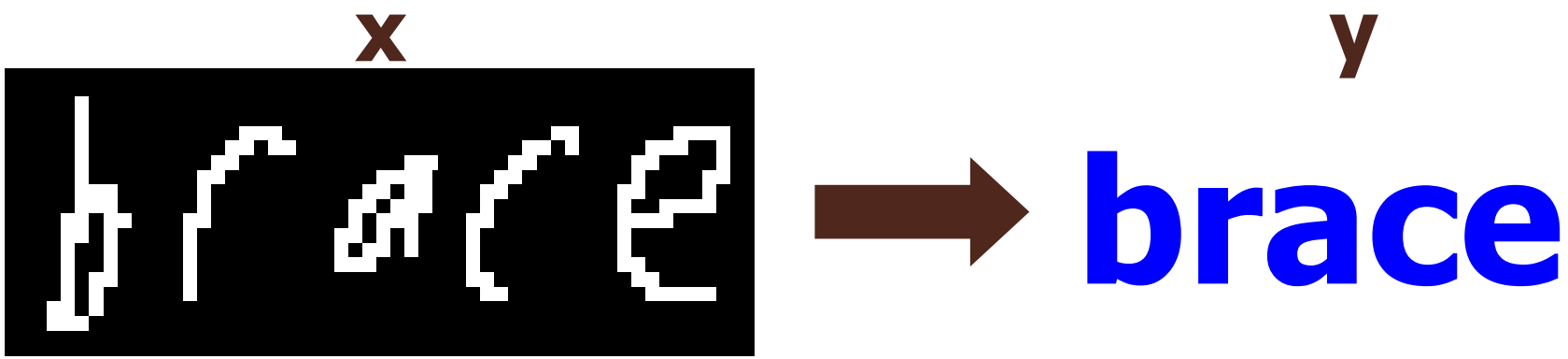
- Jointly segmenting/annotating images
- Image-image matching, image-text matching
- Problem:
  - Given structure (feature), learning $\vec{\theta}$
  - Learning sparse, interpretable, **predictive** structures/features

Dependency parsing of Sentences

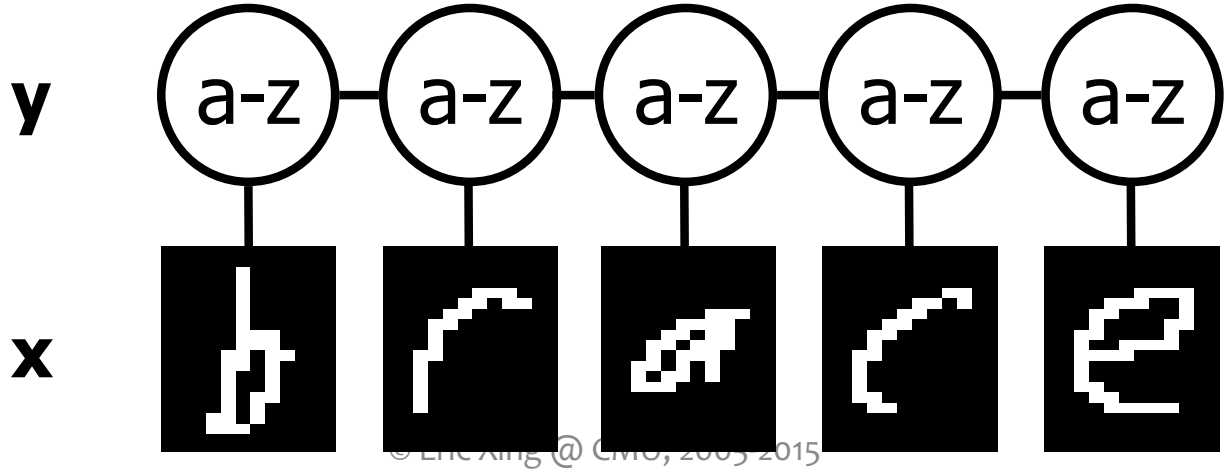


Challenge:
Structured outputs, and globally constrained to be a valid tree

OCR example



Sequential structure

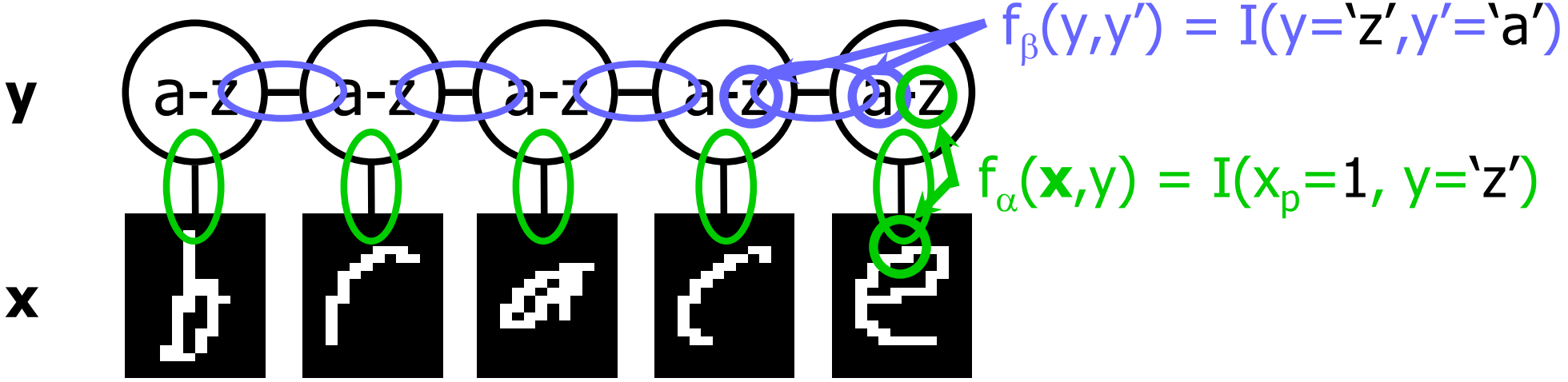


Linear-chain CRF for OCR

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_i \phi(\mathbf{x}_i, y_i) \prod_i \phi(y_i, y_{i+1})$$

$$\phi(\mathbf{x}_i, y_i) = \exp\{\sum_\alpha w_\alpha f_\alpha(\mathbf{x}_i, y_i)\}$$

$$\phi(y_i, y_{i+1}) = \exp\{\sum_\beta w_\beta f_\beta(y_i, y_{i+1})\}$$

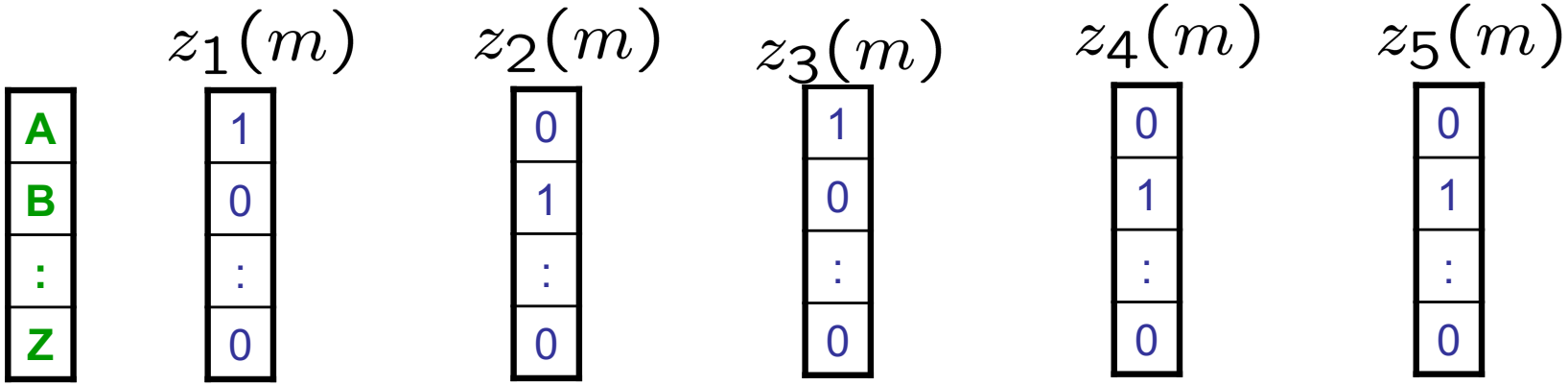


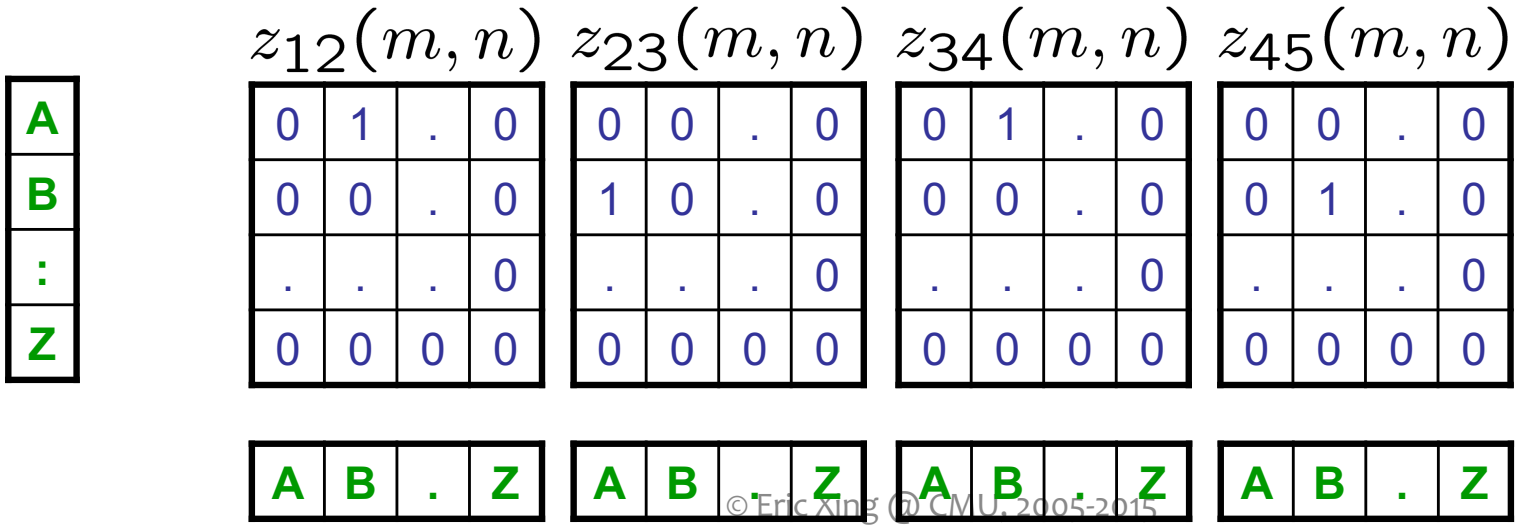
*Lafferty et al. 01

$y \Rightarrow z$ map for linear chain structures

OCR example: $\mathbf{y}$ = 'ABABB';

$\mathbf{z}$'s are the indicator variables for the corresponding classes (alphabet)





y ⇒ z map for linear chain structures

$$\max_{\mathbf{y}} \sum_j \mathbf{w}^T f_{\text{node}}(x_j, y_j) + \sum_{j,k} \mathbf{w}^T f_{\text{edge}}(\mathbf{x}_{jk}, y_j, y_k)$$

Rewriting the maximization function in terms of indicator variables:

$$\max_{\mathbf{z}} \sum_{j,m} z_j(m) \left[\mathbf{w}^\top \mathbf{f}_{\text{node}}(\mathbf{x}_j, m) \right] + \sum_{jk,m,n} z_{jk}(m,n) \left[\mathbf{w}^\top \mathbf{f}_{\text{edge}}(\mathbf{x}_{jk}, m, n) \right]$$

$z_k(n)$

0	1	0	0

$z_j(m)$

0
0
1
0

$z_{jk}(m,n)$

0	0	0	0
0	0	0	0
0	1	0	0
0	0	0	0

$$z_j(m) \geq 0; \; z_{jk}(m,n) \geq 0;$$

normalization $\sum_m z_j(m) = 1$

agreement $\sum_n z_{jk}(m,n) = z_j(m)$

integer $z_j(m) \in \mathcal{Z}, \; z_{jk}(m,n) \in \mathcal{Z}$

y ⇒ z map for linear chain structures

$$\max_{\mathbf{y}} \sum_j \mathbf{w}^T f_{\text{node}}(x_j, y_j) + \sum_{j,k} \mathbf{w}^T f_{\text{edge}}(\mathbf{x}_{jk}, y_j, y_k)$$

Rewriting the maximization function in terms of indicator variables:

$$\max_{\mathbf{z}} \sum_{j,m} z_j(m) \left[\mathbf{w}^\top \mathbf{f}_{\text{node}}(\mathbf{x}_j, m)\right] + \sum_{jk,m,n} z_{jk}(m,n) \left[\mathbf{w}^\top \mathbf{f}_{\text{edge}}(\mathbf{x}_{jk}, m, n)\right] \quad \Big\} (\mathbf{F}^\top \mathbf{w})^\top \mathbf{z}$$

$z_k(n)$

$z_j(m)$

0	1	0	0

0
0
1
0

0	0	0	0
0	0	0	0
0	1	0	0
0	0	0	0

$z_{jk}(m,n)$

$$z_j(m) \geq 0;\ z_{jk}(m,n) \geq 0;$$

normalization $\sum_m z_j(m) = 1$

agreement $\sum_n z_{jk}(m,n) = z_j(m)$

$\Big\} \mathbf{Az} = \mathbf{b}$

$$\max_{A\mathbf{z}=\mathbf{b}} (\mathbf{F}^\top \mathbf{w})^\top \mathbf{z}$$

MAP Inference

Suppose we want to predict the highest likelihood structure y, given observations x and parameters w.

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \log p_w(y|x)$$

$$= \underset{\mathbf{y}}{\operatorname{argmax}} \sum_j \mathbf{w}^T f_{\text{node}}(x_j, y_j) + \sum_{j,k} \mathbf{w}^T f_{\text{edge}}(\mathbf{x}_{jk}, y_j, y_k)$$

Idea:

1. Reformulate the problem as an integer linear program (ILP) – note that this is just going to be a new way of writing down the problem: y → z
2. Then remove the integer constraints (i.e. solve the linear program (LP) relaxation)

Lemma: (Wainwright et al., 2002) If there is a unique MAP assignment, the LP relaxation of the ILP above is guaranteed to have an integer solution, which is exactly the MAP solution!

Q&A

Q: Will MILP converge to the true MAP assignment?

A: Yes! But it might take a while. More generally, it returns a certificate of epsilon optimality, meaning we know that we are within epsilon of optimal.

Q: When is MILP for MAP inference efficient?

A: Hard to say. MILP is, in general, solving an NP-Hard problem. Its efficiency often depends on how tight the LP relaxation is.

Q&A

Q: So can we use off-the-shelf MILP solvers to do MAP inference?

A: Yes, this is exactly the point. Gurobi and CPLEX are the two main commercial options. But there are good open source options as well.

Q: How do we come up with relaxations other than the LP relaxation?

A: Dual-decomposition offers a different way of coming up with relaxations that sometimes incorporate specialized dynamic programming algorithms as well!

STRUCTURED PERCEPTRON

Linear Models

Setup: training examples are (x, y) where $\vec{x} \in \mathbb{R}^p$ $y \in \{1, \ldots, k\}$

Model: parameters $\theta \in \mathbb{R}^M$
feature function $f(x, y) \in \mathbb{R}^M$

Predict: $\hat{y} = h_\theta(x) = \underset{y \in \{1,\ldots,k\}}{\operatorname{argmax}} \; \theta^T f(\vec{x}, y)$

Ex#1: $f(\vec{x}, y) = \text{vectorize}\left(\begin{bmatrix} 0 & 0 & \cdots & 0 \\ \vdots & & & \\ x_1 & x_2 & \cdots & x_p \\ \vdots & & & \\ 0 & 0 & \cdots & 0 \end{bmatrix} \right)$

$K \times p$ matrix

1
2
⋮
y
⋮
k-1
k

only the y^{th} row is nonzero

$= [0\,0 \cdots 0\; x_1\; x_2 \cdots x_p\; 0\,0 \cdots 0]^T$

$\Rightarrow M = K \times p$

Linear Models

Setting: training examples are (x, y) where $\vec{x} \in \mathbb{R}^p$ $y \in \{1, \ldots, k\}$

Model: parameters $\theta \in \mathbb{R}^M$
feature function $f(x, y) \in \mathbb{R}^M$

Predict: $\hat{y} = h_\theta(x) = \underset{y \in \{1,\ldots,k\}}{\operatorname{argmax}} \theta^T f(\vec{x}, y)$

Ex #2: Suppose $\{1, \ldots, k\}$ exist in some hierarchy and $\vec{x} \in \{0,1\}^p$

$$f(\vec{x}, y) = \begin{bmatrix} \mathbb{1}(x_1 = 1 \text{ and } y \in \text{mammal}) \\ \mathbb{1}(x_1 = 1 \text{ and } y = \text{cat}) \\ \mathbb{1}(x_1 = 1 \text{ and } y = \text{dog}) \\ \vdots \\ \mathbb{1}(x_p = 1 \text{ and } y \in \text{amph.}) \\ \vdots \end{bmatrix}$$

Structured Perceptron

Whiteboard

- Multiclass Perceptron
- Structured Perceptron

Structured Perceptron

Mistake Bound:

Definition 1 *Let* $\overline{\mathbf{GEN}}(x_i) = \mathbf{GEN}(x_i) - \{y_i\}$. *In other words* $\overline{\mathbf{GEN}}(x_i)$ *is the set of* incorrect *candidates for an example* x_i. *We will say that a training sequence* (x_i, y_i) *for* $i = 1 \ldots n$ *is* **separable with margin** $\delta > 0$ *if there exists some vector* $\mathbf{U}$ *with* $||\mathbf{U}|| = 1$ *such that*

$$\forall i, \forall z \in \overline{\mathbf{GEN}}(x_i), \quad \mathbf{U} \cdot \Phi(x_i, y_i) - \mathbf{U} \cdot \Phi(x_i, z) \geq \delta \quad (3)$$

($||\mathbf{U}||$ *is the 2-norm of* $\mathbf{U}$, *i.e.,* $||\mathbf{U}|| = \sqrt{\sum_s \mathbf{U}_s^2}$.*)*

Theorem 1 *For any training sequence* (x_i, y_i) *which is separable with margin* δ, *then for the perceptron algorithm in figure 2*

$$\textit{Number of mistakes} \leq \frac{R^2}{\delta^2}$$

where R *is a constant such that* $\forall i, \forall z \in \overline{\mathbf{GEN}}(x_i) \quad ||\Phi(x_i, y_i) - \Phi(x_i, z)|| \leq R$.

from Collins (2002)

Structured Perceptron

- Results from Collins (2002) on two **sequence tagging** problems
- Metrics:
  - **F-measure:** higher is better
  - **Error:** lower is better
- Comparison of…
  - Structured Perceptron **with** and **without** averaging
  - Maximum entropy Markov model (**ME**MM)
- Takeaways:
  - incredibly **easy to implement**
  - typically **blazing fast**

NP Chunking Results

Method	F-Measure	Numits
Perc, avg, cc=0	93.53	13
Perc, noavg, cc=0	93.04	35
Perc, avg, cc=5	93.33	9
Perc, noavg, cc=5	91.88	39
ME, cc=0	92.34	900
ME, cc=5	92.65	200

POS Tagging Results

Method	Error rate/%	Numits
Perc, avg, cc=0	2.93	10
Perc, noavg, cc=0	3.68	20
Perc, avg, cc=5	3.03	6
Perc, noavg, cc=5	4.04	17
ME, cc=0	3.4	100
ME, cc=5	3.28	200

Figure 4: Results for various methods on the part-of-speech tagging and chunking tasks on development data. All scores are error percentages. Numits is the number of training iterations at which the best score is achieved. Perc is the perceptron algorithm, ME is the maximum entropy method. Avg/noavg is the perceptron with or without averaged parameter vectors. cc=5 means only features occurring 5 times or more in training are included, cc=0 means all features in training are included.

Table from Collins (2002)

aka. Max-Margin Markov Networks (M^3Ns)

STRUCTURED SVM

Support Vector Machines

Binary SVM

Data: $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ where $\vec{x}^{(i)} \in \mathbb{R}^M$ $y^{(i)} \in \{+1, -1\}$

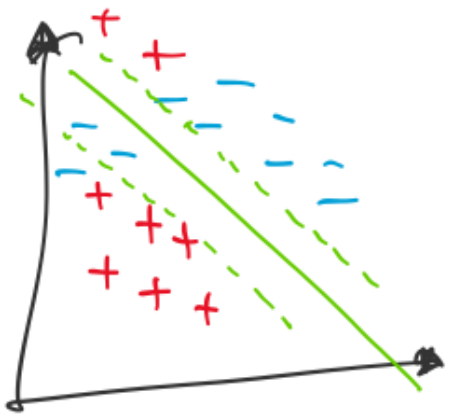
Model: $\hat{y} = h_{w,b}(x) = \text{sign}(w^T x^{(i)} + b)$

Quadratic Program (QP): (hyperparameter)

$$\min_{w,e} \frac{1}{2}(\|w\|_2)^2 + C\sum_{i=1}^{N} e_i$$

$$\text{s.t.} \quad y^{(i)}(w^T x^{(i)} + b) \geq 1 - e_i \quad \forall i$$

$$e_i \geq 0 \quad \forall i$$



Binary SVM: Hinge Loss

Hinge Loss: $\ell^{\text{hinge}}(y, \hat{y}) = \max(0, 1 - y\hat{y})$

Unconstrained Opt. Problem:

Observe: $e_i \geq 1 - y^{(i)}\underbrace{(w^T x^{(i)} + b)}_{\hat{y}}$ and $e_i \geq 0$ $\Rightarrow e_i \geq \max(0, 1 - y^{(i)}(w^T x^{(i)} + b))$

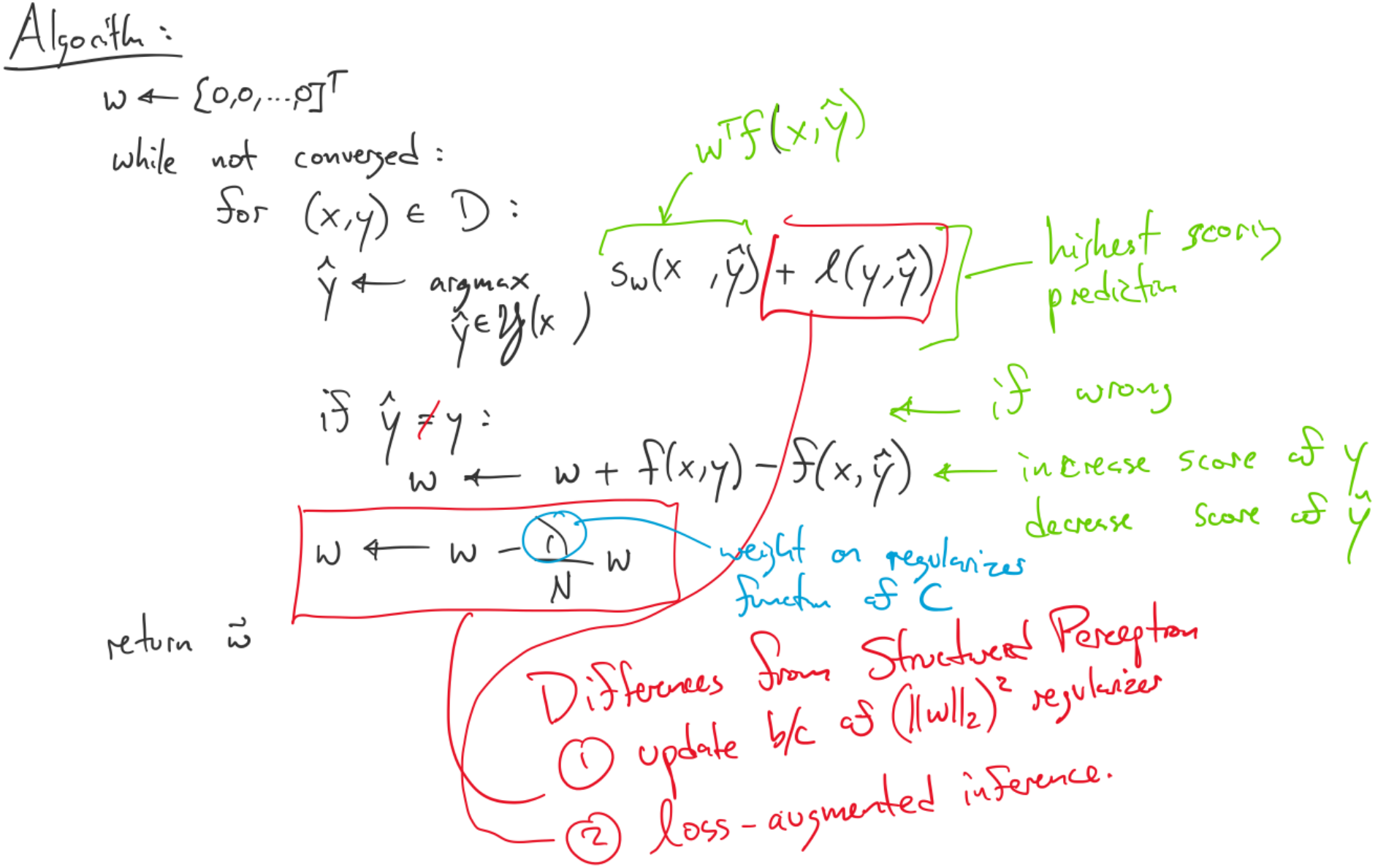
$$\min_{w} \underbrace{\frac{1}{2}(\|w\|_2)^2}_{\text{large margin}} + C\underbrace{\sum_{i=1}^{N} \max(0, 1 - y^{(i)}(w^T x^{(i)} + b))}_{\text{few/small errors}}$$

Structured SVM

Whiteboard

- Structured Large Margin
- Structured Hinge Loss
- Gradient of Structured Hinge Loss
- SGD for Structured SVM

SGD for Structured SVM



Loss-Augmented Inference

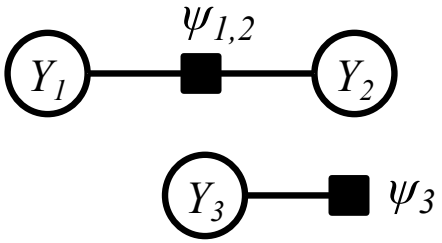
The Loss Augmented Inference problem is defined as:

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \log p(\mathbf{y}) + \ell(\mathbf{y}, \mathbf{y}^*)$$

- We can only solve this problem at training time, since it involves **y***.
- The goal is to identify the **y** with score **plus** loss
- This **y** corresponds to the one that most violates the loss-scaled-margin
- This is exactly what we need to train with Structured SVM

Loss-Augmented Inference

Example



Joint Distribution:

$$\log p(\mathbf{y}) = \log \psi_{12}(y_1, y_2) + \log \psi_3(y_3) - \log Z$$

where $y_t \in$ {red, blue, purple}

Loss Function:

$$\ell(\hat{\mathbf{y}}, \mathbf{y}^*) = \ell_{12}(\hat{y}_1, \hat{y}_2, y_1^*, y_2^*) + \ell_3(\hat{y}_3, y_3^*)$$

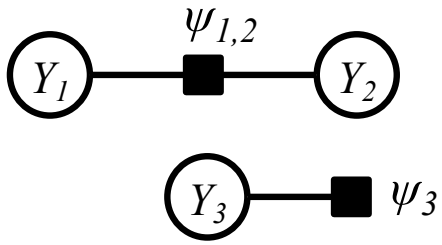
where

$$\ell_{12}(\hat{y}_1, \hat{y}_2, y_1^*, y_2^*) = \begin{cases} 100 \text{ if } y_1^* = y^*2 = \text{purple} \neq \hat{y}_1 \neq \hat{y}_2 \\ \mathbb{1}(y_1^* \neq \hat{y}_1) + \mathbb{1}(y_2^* \neq \hat{y}_2) \text{ otherwise} \end{cases}$$

$$\ell_3(\hat{y}_3, y_3^*) = \mathbb{1}(y_3^* = \hat{y}_3$$

Loss-Augmented Inference

Example



Joint Distribution:

$$\log p(\mathbf{y}) = \log \psi_{12}(y_1, y_2) + \log \psi_3(y_3) - \log Z$$

Loss Function:

$$\ell(\hat{\mathbf{y}}, \mathbf{y}^*) = \ell_{12}(\hat{y}_1, \hat{y}_2, y_1^*, y_2^*) + \ell_3(\hat{y}_3, y_3^*)$$

Loss Augmented Inference:

1. Given $\mathbf{y}^*$, define a new factor graph

$$\log p'(\mathbf{y}) = \log \psi'_{12}(y_1, y_2) + \log \psi'_3(y_3) - \log Z'$$

$$\text{where } \log \psi'_{12}(y_1, y_2) = \log \psi_{12}(y_1, y_2) + \ell_{12}(\hat{y}_1, \hat{y}_2, y_1^*, y_2^*)$$

$$\log \psi'_3(y_3) = \log \psi_3(y_3) + \ell_3(\hat{y}_3, y_3^*)$$

2. Run MAP inference on new F.G. to solve the L.A.I problem

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}} \log p(\mathbf{y}) + \ell(\mathbf{y}, \mathbf{y}^*)$$

$$= \operatorname*{argmax}_{\mathbf{y}} \log p'(\mathbf{y})$$

This equality holds true because of our carefully defined new factor graph