Part 2:
Structured Learning

Kai-Wei Chang

Microsoft Research → Univ. of Virginia

- Part 2: **Learning a Structured Prediction Model** (45min)

  - **Definition of structured learning**
  - **Local Learning v.s. Global Learning**
  - **Global Learning Algorithms**
    - Online learning: Structured Perceptron
    - Batch learning: Structured SVM
  - **Optimization methods for Structured SVM**
    - Stochastic Gradient Decent
    - Dual Coordinate Descent
    - Learning on a multi-core machine

CCM Formulations

$$\mathbf{y} = \boldsymbol{argmax}_{y \in Y} \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) + \boldsymbol{u}^T \boldsymbol{C}(\boldsymbol{x}, \boldsymbol{y})$$

This part of the tutorial focuses on learning **w** (and **u**)

Learning a Structured Prediction Model

- Input: $x \in X$
- Truth: $y^* \in Y(x)$
- Predicted: $h(x) \in Y(x)$
- Loss: $\Delta(y, y^*)$

I	can	can	a	can
Pro	Md	Vb	Dt	Nn
Pro	Md	Md	Dt	Vb
Pro	Md	Md	Dt	Nn
Pro	Md	Nn	Dt	Md
Pro	Md	Nn	Dt	Vb

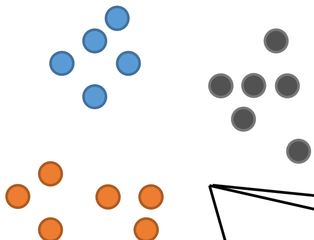
Goal: find $h \in H$ such that $h(\mathrm{x}) \in Y(X)$ minimizing $E_{(x,y)\sim D}\left[\Delta\left(y, h(x)\right)\right]$ based on N samples $(x_n, y_n) \sim D$

Learning Paradigms [Punyakanok+ 05]

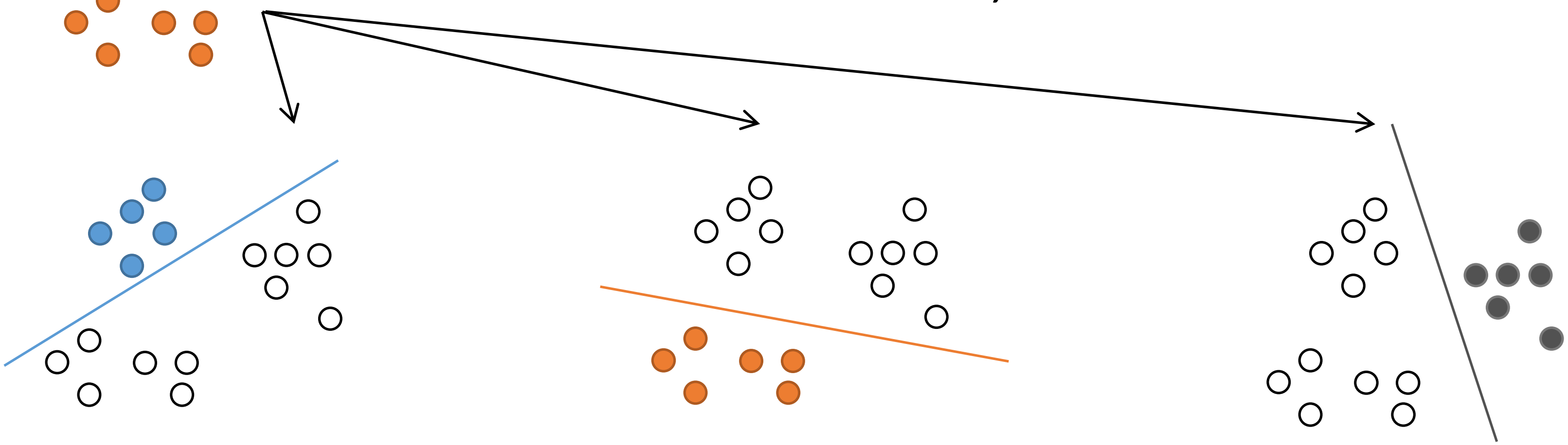
- Local Learning
  - Learning local models independently
  - Ensure output is coherent at test time
  - E.g., One-against-all multiclass classification

- Global Learning
  - Learning with inference
  - Training and testing are consistent
  - Constrained classification for multiclass
    [Har-Peled et. el 2002; Crammer et. al 2002]

Visualizing One-vs-all



From the full dataset, construct three binary classifiers, one for each class



$\boxed{\mathbf{w}_{\mathbf{blue}}{}^{T}\mathbf{x}} > 0$ for **blue inputs**

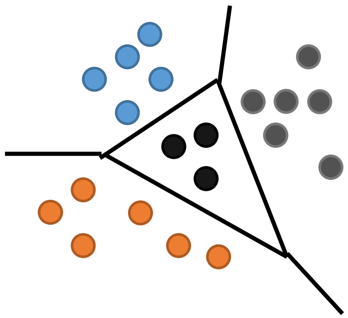
$\mathbf{w}_{\mathbf{red}}{}^{T}\mathbf{x} > 0$ for **red inputs**

$\mathbf{w}_{\mathbf{green}}{}^{T}\mathbf{x} > 0$ for **green inputs**

Notation: Score for blue label

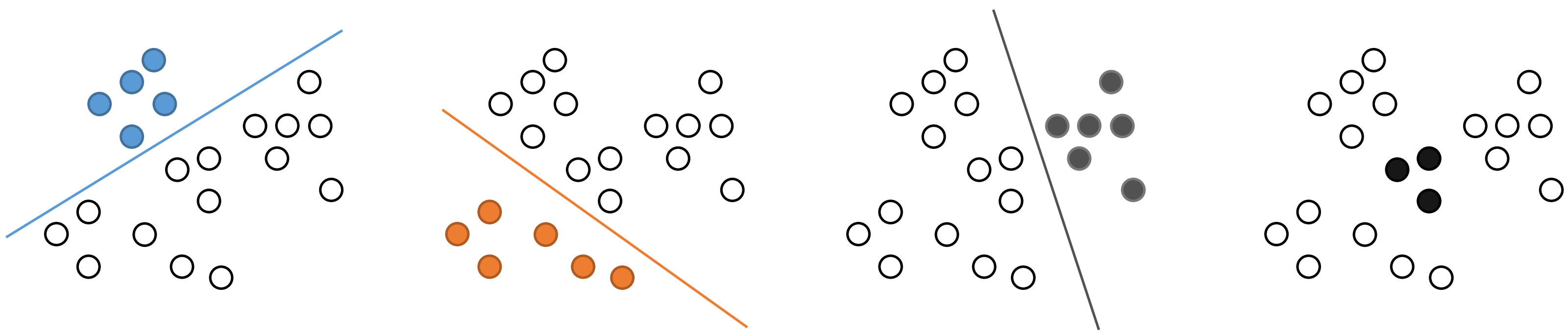
Winner Take All will predict the right answer. Only the correct label will have a positive score

One-vs-all may not always work



Black points are not separable with a single binary classifier

The decomposition will not work for these cases!



$\mathbf{w}_{\mathbf{blue}}^T\mathbf{x} > 0$ for **blue** **inputs**

$\mathbf{w}_{\mathbf{red}}^T\mathbf{x} > 0$ for **red** **inputs**

$\mathbf{w}_{\mathbf{green}}^T\mathbf{x} > 0$ for **green** **inputs**

???

Local Learning: One-vs-all classification

- Easy to learn
  - Use any binary classifier learning algorithm
- Problems
  - Calibration issues
    - We are comparing scores produced by K classifiers trained independently. No reason for the scores to be in the same numerical range!
  - Might not always work
    - Yet, works fairly well in many cases, especially if the underlying binary classifiers are tuned, regularized

Global Learning Motivation

- Decomposition methods
  - Do not account for how the final predictor will be used
  - Do not optimize any global measure of correctness

- Goal: To train a multiclass classifier that is "global"

Global "One-vs-all" Approach

- *Idea:* Create K classifiers $\mathbf{w}_1$, $\mathbf{w}_2$, …, $\mathbf{w}_K$.
  For examples with label i, we want

$$\boldsymbol{w}_i^T\boldsymbol{x} > \boldsymbol{w}_j^T\boldsymbol{x} \textit{ for all } j$$

- Prediction: $\text{argmax}_i\ \mathbf{w}_i^T\mathbf{x}$
- Training: For each training example $(\boldsymbol{x}_i, \boldsymbol{y}_i)$:

$$\hat{y} \leftarrow arg\max_{\boldsymbol{j}} \boldsymbol{w}_{\boldsymbol{j}}^T \phi(\boldsymbol{x}_i, y_i)$$

if $\hat{y} \neq y_i$ η: learning rate

$$\boldsymbol{w}_{y_i} \leftarrow \boldsymbol{w}_{y_i} + \eta\boldsymbol{x}_i \qquad \text{(promote)}$$

$$\boldsymbol{w}_{\hat{y}} \leftarrow \boldsymbol{w}_{\hat{y}} - \eta\boldsymbol{x}_i \qquad \text{(demote)}$$

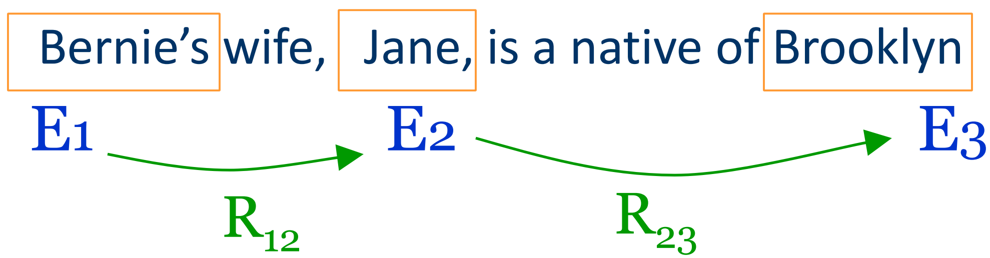
Joint Inference with General Constraint Structure
[Roth&Yih'04,07,….]

Recognizing Entities and Relations

other	0.05
per	**0.85**
loc	0.10

other	0.10
per	**0.60**
loc	0.30

other	0.05
per	0.50
loc	**0.45**



irrelevant	0.05
spouse_of	**0.45**
born_in	0.50

irrelevant	0.10
spouse_of	0.05
born_in	**0.85**

Models could be learned separately/jointly; constraints may come up only at decision time.

Training Methods

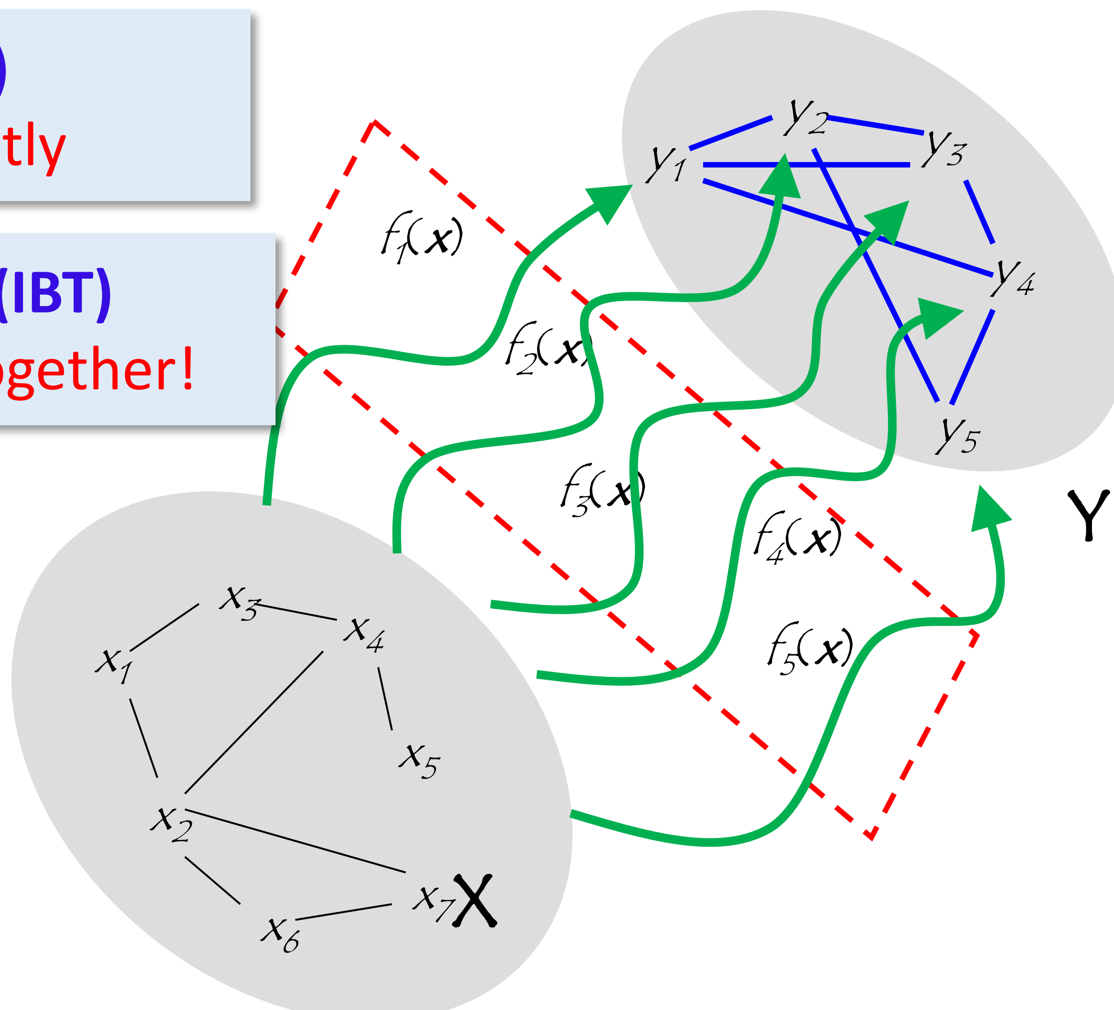
Learning + Inference (L+I)

Learn models independently

Inference Based Training (IBT)

Learn one model, all y's together!

Intuition: Learning with constraints may make learning more difficult



Local Learning v.s. Global Learning

- Advantages of local training:
  - Modular, very simple to implement
  - Often efficient

- Advantages of global training:
  - Models directly capture correlations between outputs (have better theoretical guarantees)
  - Might achieve better performance when data is sufficient

Structured Prediction: Learning

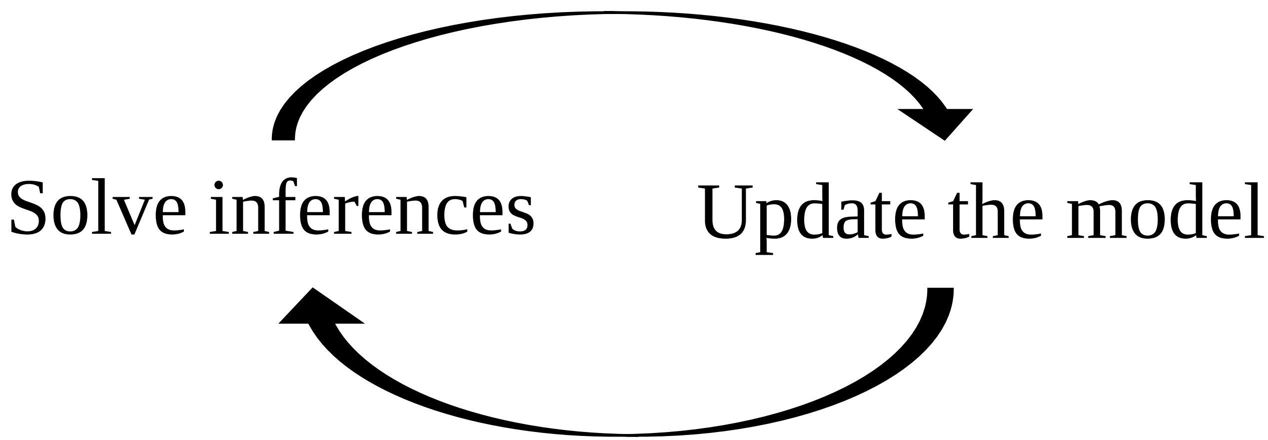
- Learning is thus driven by the attempt to find a weight vector w such that for each given annotated example (x_i, y_i):

$$\text{Score of annotated structure} \geq \text{Score of any other structure} + \text{Penalty for predicting other structure}$$

- The update of the weight vector w can be done in an on-line or a batch fashion

Global Learning Algorithms

- Online e.g., Structured Perceptron [Collins 02]
  - Receive an instance; and update
- Batch e.g., Structured SVM [Taskar+05]
  - Collect a set of data; formulate learning as an optimization problem



Structured Perceptron

- Perceptron (binary classification):

For each training example $(\boldsymbol{x_i}, y_i)$:

$$\hat{y} \leftarrow sgn(\boldsymbol{w}^T \boldsymbol{x_i})$$

if $y_i \neq \hat{y}$, then $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta y_i \boldsymbol{x_i}$

Perceptron is a special case when

$$\phi(\boldsymbol{x}, y) = y\boldsymbol{x}$$

$$\eta' = \frac{1}{2}\eta$$

- Structured Perceptron

For each training example $(\boldsymbol{x_i}, \boldsymbol{y_i})$:

$$\hat{\boldsymbol{y}} \leftarrow arg \max_{\boldsymbol{y} \in Y} \boldsymbol{w}^T \phi(\boldsymbol{x}_i, \boldsymbol{y_i})$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta' \left[\phi(\boldsymbol{x}_i, \boldsymbol{y}_i) - \phi(\boldsymbol{x}_i, \hat{\boldsymbol{y}})\right]$$

Structured Perceptron Variations

- (Marginal) Structured Perceptron

  For each training example $(\boldsymbol{x_i}, \boldsymbol{y_i})$:

$$\hat{\boldsymbol{y}} \leftarrow \ arg \max_{\boldsymbol{y} \in Y} \boldsymbol{w}^T \phi(\boldsymbol{x}_i, \boldsymbol{y_i}) + \Delta(\mathbf{y}, \mathbf{y}_\mathrm{i})$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta' \left[\phi(\boldsymbol{x}_i, \boldsymbol{y}_i) - \phi(\boldsymbol{x}_i, \hat{\boldsymbol{y}}) \right]$$

- Parallel Structured Perceptron [McDonald et al 10]:

1. Split data into $\boldsymbol{p}$ parts.
2. Train Structured Perceptron on each data block in parallel.
3. Mixed the models using a linear combination.
4. Repeat Step 2 and use the mixed model as the initial model.

Recap

- Learning is thus driven by the attempt to find a weight vector w such that for each given annotated example (x_i, y_i):

Score of annotated structure $\geq$ **Score of any other structure** $+$ **Penalty for predicting other structure**

- Introduce slack variables $\{\xi_i\}$

$$\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}_i) - \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}) \geq \boldsymbol{\Delta}(\mathbf{y}_\mathrm{i}, \mathbf{y}) - \xi_i$$

Structured SVM (Batch)

(Tsochantaridis et al. 05)

For all samples and feasible structures

Given a set of training examples $D = \{\boldsymbol{x_i}, \boldsymbol{y_i}\}_{i=1}^{l}$

$$\min_{\boldsymbol{w},\xi} \quad \frac{1}{2}\boldsymbol{w^T w} + \boldsymbol{C}\sum_i \xi_i$$

$$\boldsymbol{s.t.} \quad \boldsymbol{w^T \phi(x_i, y_i)} - \boldsymbol{w^T \phi(x_i, y)} \geq \boldsymbol{\Delta(y_i, y)} - \xi_i \quad \boxed{\forall \boldsymbol{i, y} \in \boldsymbol{Y_i}}$$

Note:

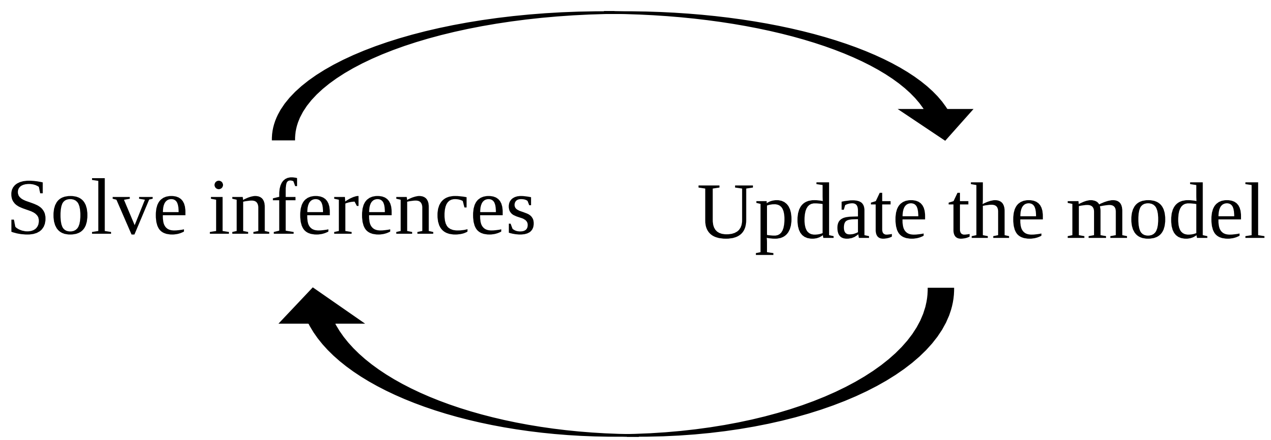
1. $\boldsymbol{w^T \phi(x, y)}$ is the **scoring function** used in inference.
2. Equivalent to the following optimization problem:

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w^T w} + \boldsymbol{C}\sum_i \left(\max_{\boldsymbol{y}} \left[\boldsymbol{\Delta(y_i, y)} + \boldsymbol{w^T \phi(x_i, y)}\right] - \boldsymbol{w^T \phi(x_i, y_i)} \right)$$

Loss-augmented inference

Optimization Methods for Structured SVM

- Online e.g., Structured Perceptron [Collins 02]
- Batch e.g., Structured SVM [Taskar+05]
  - Cutting plane: [Tsochantaridis+ 05, Joachims+ 09]
  - Dual Coordinate Descent: [Shevade+ 11, Chang+ 13]
  - Block-Coordinate Frank-Wolfe: [Lacoste-Julien+ 13]
  - Parallel Dual Coordinate Descent: [Chang+ 13a]



Stochastic (sub-)gradient descent

To minimize a function $\mathrm{g}(z)$ that has the form $\sum_i g_i(z)$

- Initialize z_0
- Iterate until convergence
  - Pick a random g_i and compute its (sub)gradient at z_t: $\nabla g_i(z_t)$
  - Update: $z_{t+1} \leftarrow z_t - \gamma_t \nabla g_i(z_t)$

γ_t: learning rate

General idea: Replace the gradient with a noisy estimate

Sub-gradient computation

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \boldsymbol{C}\sum_i \left(\max_{\boldsymbol{y}} \left[\boldsymbol{\Delta}(\mathbf{y_i}, \mathbf{y}) + \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}) \right] - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}_i) \right)$$

- Solve the max. Suppose solution is y'
  - The loss-augmented/loss-sensitive/cost-augmented inference step
- Compute gradient of

$$\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \boldsymbol{C}\left(\left[\boldsymbol{\Delta}(\mathbf{y_i}, \mathbf{y}) + \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}) \right] - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}_i) \right)$$

- Subgradient is

$$\mathbf{w} + C\left(\phi(\mathbf{x}_i, \mathbf{y}') - \phi(\mathbf{x}_i, \mathbf{y}_i)\right)$$

SGD for structural SVM: The update

- Solve inference and compute sub-gradient:

$$\mathbf{w} + C\left(\phi(\mathbf{x}_i, \mathbf{y}') - \phi(\mathbf{x}_i, \mathbf{y}_i)\right)$$

- At each step, go down the gradient:

$$\mathbf{w} \leftarrow \mathbf{w} - \gamma_t\left(\mathbf{w} + C\left(\phi(\mathbf{x}_i, \mathbf{y}') - \phi(\mathbf{x}_i, \mathbf{y}_i)\right)\right)$$

- Compared to Structured Perception update:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta' \left[\phi(\boldsymbol{x}_i, \boldsymbol{y}_i) - \phi(\boldsymbol{x}_i, \widehat{\boldsymbol{y}})\right]$$

($\boldsymbol{\gamma_t}, \eta$: learning rate)

L2-loss Structured SVM

Given a set of training examples $D = \{\boldsymbol{x_i}, \boldsymbol{y_i}\}_{i=1}^{l}$

$$\min_{\boldsymbol{w},\xi} \quad \frac{\mathbf{1}}{\mathbf{2}}\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w} + \boldsymbol{C}\sum_{\boldsymbol{i}}\boldsymbol{\xi}_{\boldsymbol{i}}^{\mathbf{2}}$$

$$\boldsymbol{s.t.} \quad \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{\phi}(\boldsymbol{x_i}, \boldsymbol{y_i}) - \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{\phi}(\boldsymbol{x_i}, \boldsymbol{y}) \geq \boldsymbol{\Delta}(\mathbf{y_i}, \mathbf{y}) - \boldsymbol{\xi_i} \quad \forall \boldsymbol{i}, \boldsymbol{y} \in \boldsymbol{Y_i}$$

- ❑ Solve problem in the dual space
  - ❑ **Recall that in dual SVM: one dual variable for each instance**
  - ❑ **In structured case: one for (instance, output assignment)**

Dual Problem of Structural SVM

$$\min_{\boldsymbol{\alpha} > 0} \quad D(\boldsymbol{\alpha}), \text{ and}$$

$$D(\boldsymbol{\alpha}) \equiv \frac{1}{2} \left\| \sum_{\alpha_{i,\boldsymbol{y}}} \alpha_{i,\boldsymbol{y}} \phi(\boldsymbol{y}, \boldsymbol{y}_i, \boldsymbol{x}_i) \right\|^2 + \frac{1}{4C} \sum_i \left(\sum_{\boldsymbol{y}} \alpha_{i,\boldsymbol{y}} \right)^2 - \sum_{i,\boldsymbol{y}} \Delta(\boldsymbol{y}, \boldsymbol{y}_i) \boxed{\alpha_{i,\boldsymbol{y}}},$$

where $\phi(\boldsymbol{y}, \boldsymbol{y}_i, \boldsymbol{x}_i) = \phi(\boldsymbol{y}_i, \boldsymbol{x}_i) - \phi(\boldsymbol{y}, \boldsymbol{x}_i).$

- Number of α variables can be exponentially large.
- Relationship between w^* and α^*

Corresponds to different y.

$$\boldsymbol{w}^* = \sum_{i,\boldsymbol{y}} \alpha^*_{i,\boldsymbol{y}} \phi(\boldsymbol{y}, \boldsymbol{y}_i, \boldsymbol{x}_i).$$

For linear model: maintain the relationship between w and α though out the learning process [Hsieh et.al. 08].

Structured Learning by Dual Coordinate Descent

- Number of dual variables can be exponentially large
- Maintain an active set A of dual variables:
  - Identify dual variables that will be likely non-zero
- Single-thread implementation:
  - Select and maintain A (active set selection step).

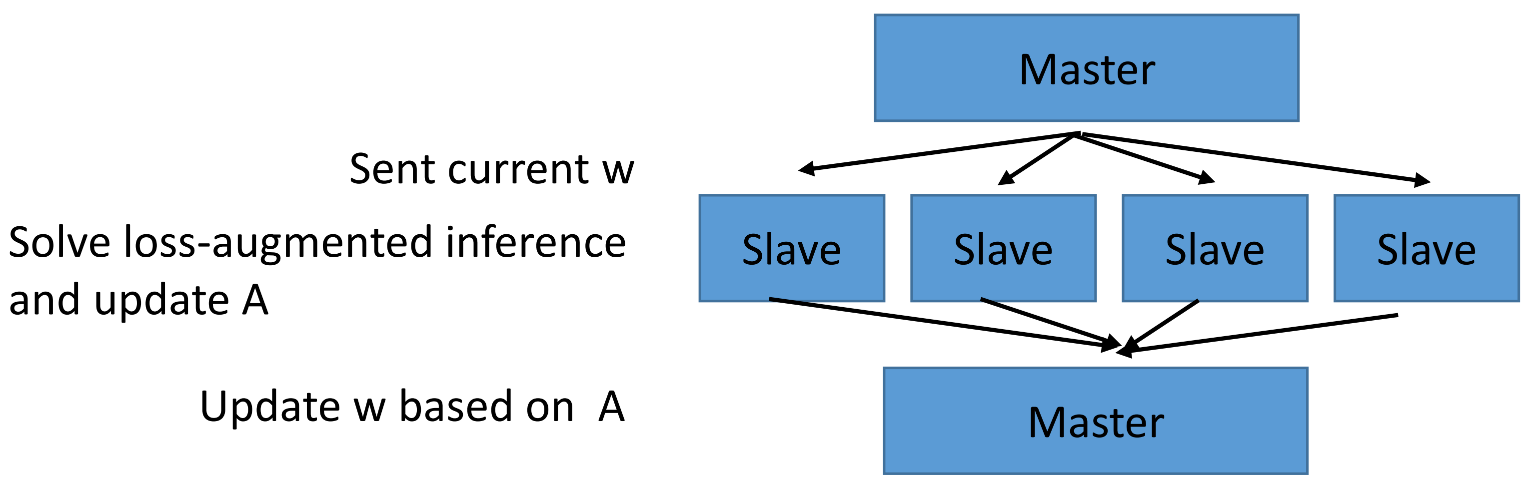
$$\max_{y \in \mathcal{Y}_i} \quad w^T \phi(x_i, y) + \Delta(y_i, y)$$

  - Update the values of $\alpha_{i,y} \in A$ (learning step).
    - (Approximately) solving a sub-problem.

A parallel Dual Coordinate Descent Algorithm

A Master-Slave architecture (MS-DCD):

- Given $\boldsymbol{p}$ processors: split data into $\boldsymbol{p}$ parts.
- At each iterations:
  - **Master sends current model to slave threads.**
  - **Each slave thread solves loss-augmented inference problems associated with a data block and updates the active set.**
  - **After all slave threads finish, master thread updates the model $\boldsymbol{w}$ according to the active set.**

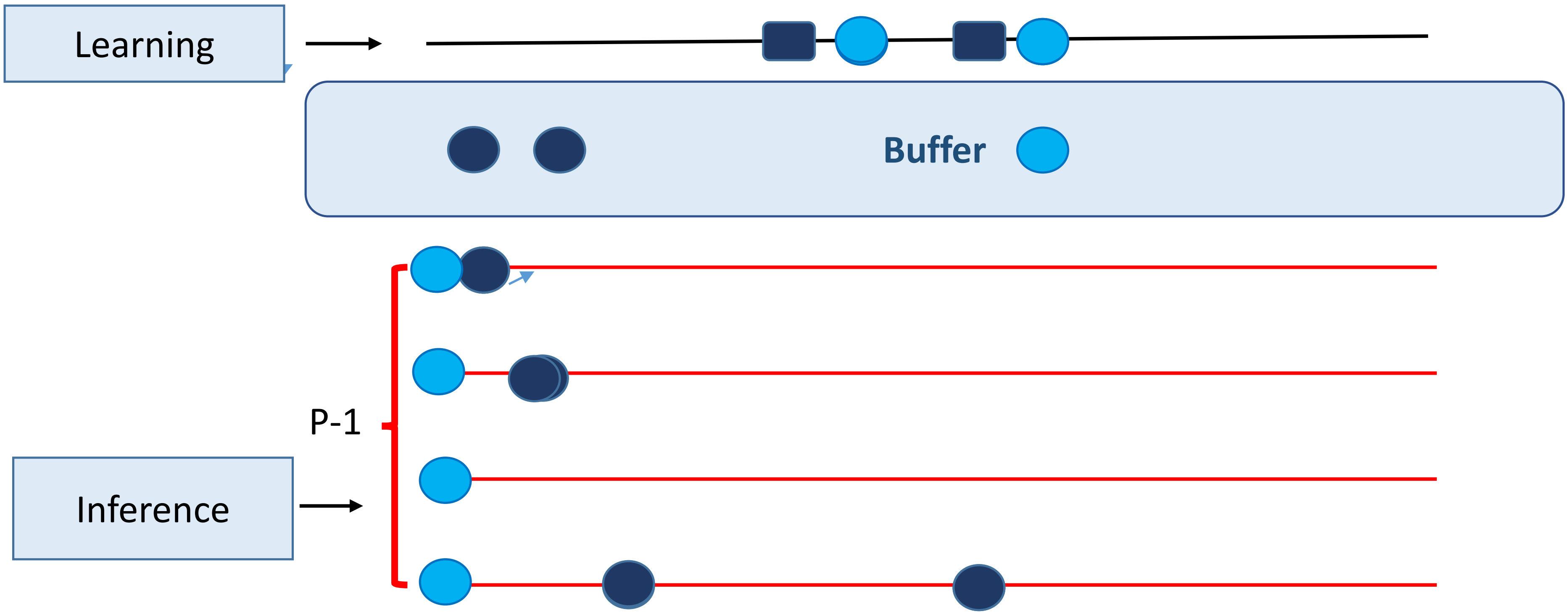


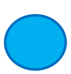
DEMI-DCD

DEMI-DCD: Decouple Model-update and Inference with Dual Coordinate Descent.

- Let $\boldsymbol{p}$ be #threads, and split training data into $\mathrm{B}_1, \mathrm{B}_2, \ldots \mathrm{B}_{\mathrm{p}-1}$.
- Active set selection thread $\boldsymbol{j}$: select and maintain the active set $\boldsymbol{A_i}$ for each example $\boldsymbol{i}$ in B_j.
- Learning thread: loop over all examples and update model w.
- $\boldsymbol{A}$ and w are shared between threads using shared memory buffers.

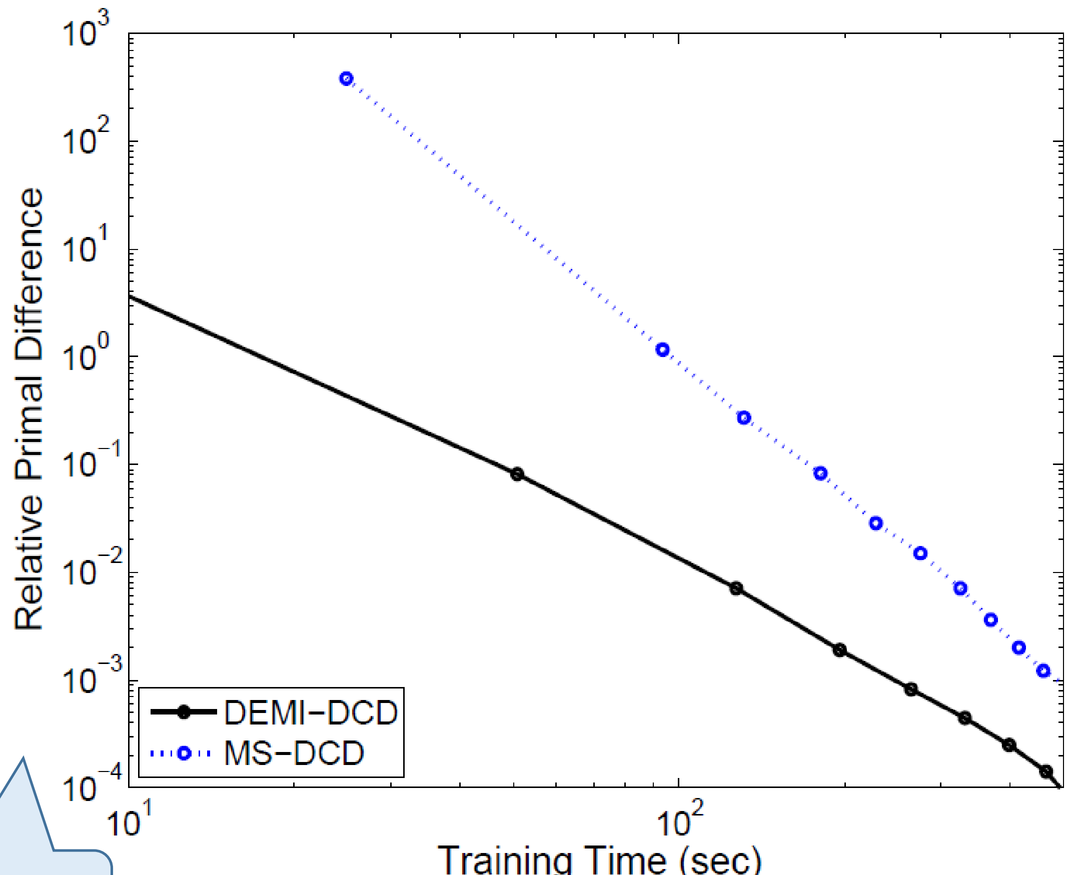
Animation




: $\{\alpha_{i,y}\}$ variables

Convergence on Primal Function Value

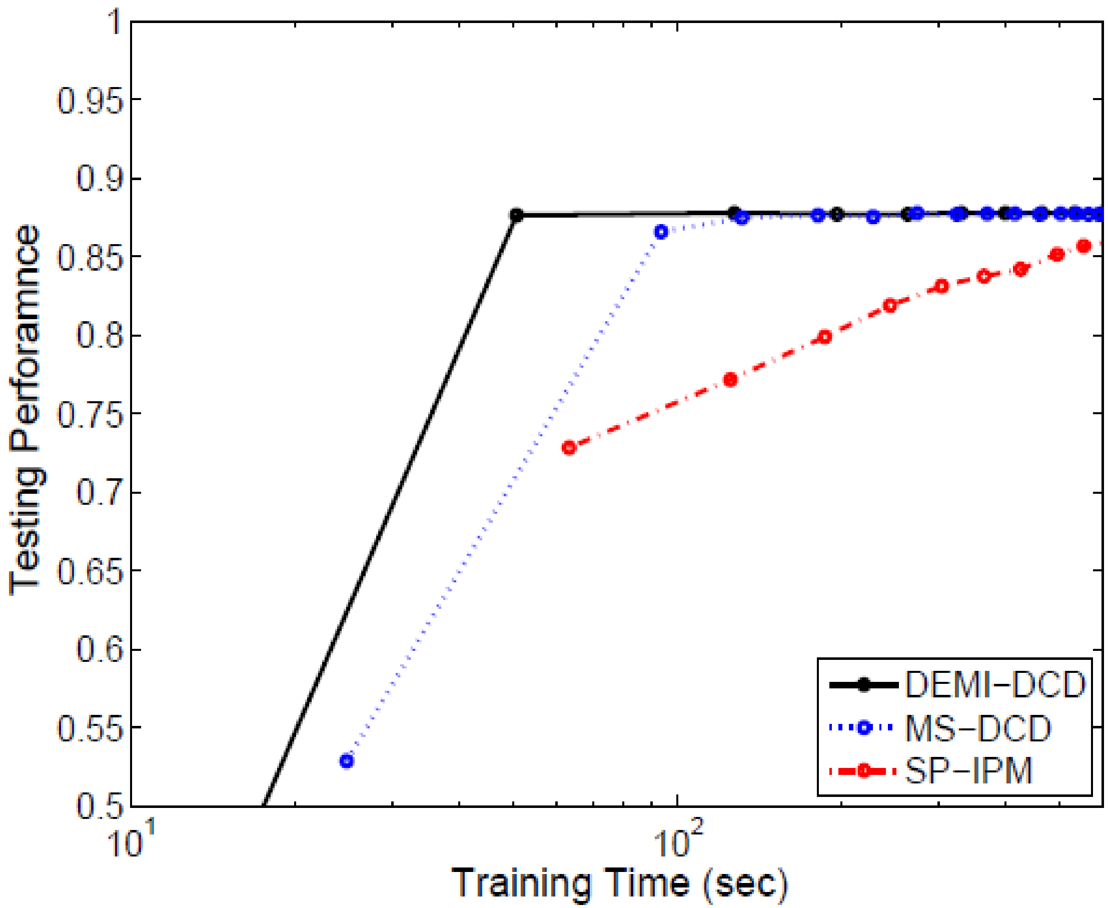
Relative primal function value difference along training time (Entity-Relation task)



Log-scale

Test Performance

Test Performance (F1 score) along training time



- Part 2: **Learning a Structured Prediction Model**
  - **Definition of structured learning**
  - **Local Learning v.s. Global Learning**
  - **Global Learning Algorithms**
    - Online learning: Structured Perceptron
    - Batch learning: Structured SVM
  - **Optimization methods for Structured SVM**
    - Stochastic Gradient Decent
    - Dual Coordinate Descent
    - Learning on a multi-core machine

In part 4, we will describe how to use them in practice