Randomized Greedy Search for Structured Prediction: Amortized Inference and Learning

Chao Ma[1], Reza Chowdhuri[2], **Aryan Deshwal[2]**, Rakibul Islam[2], Jana Doppa[2], Dan Roth[3]

[1] Oregon State University [2] Washington State University

[3] University of Pennsylvania

Motivation

- **Structured Prediction problems are very common**
  - Natural language processing
  - Computer vision
  - Computational biology
  - Planning
  - Social networks
  - ....

NLP Examples: POS Tagging and Parsing

- **POS Tagging**

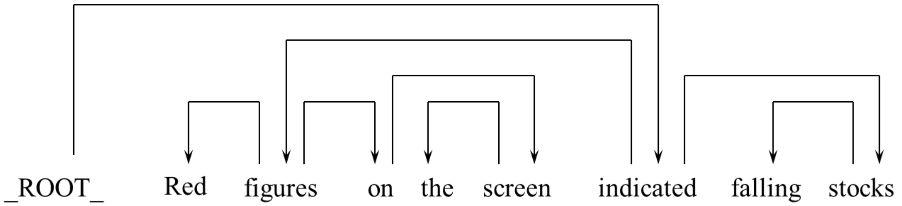
x = "The cat ran" y = *<article> <noun> <verb>*

- **Parsing**

$\boldsymbol{x}$

"Red figures on the screen indicated falling stocks"

$\boldsymbol{y}$



Computer Vision: Examples

- Handwriting Recognition

s t r u c t u r e d

- Scene Labeling

Common Theme

- POS tagging, parsing, scene labeling…
  - **Inputs and outputs are highly structured**

- Studied under a sub-field of machine learning called "**Structured Prediction**"
  - Generalization of standard classification
  - Exponential no. of classes (e.g., all POS tag sequences)

- **Key challenge for inference and learning:** large size of structured output spaces

Cost Function Learning Approaches

- Generalization of traditional ML approaches to structured outputs
  - SVMs $\Rightarrow$ Structured SVM [Tsochantaridis et al., 2004]
  - Logistic Regression $\Rightarrow$ Conditional Random Fields [Lafferty et al., 2001]
  - Perceptron $\Rightarrow$ Structured Perceptron [Collins 2002]

Cost Function Learning: Approaches

- Most algorithms learn parameters of linear models
  - $\phi(x, y)$ is n-dim feature vector over input-output pairs
  - w is n-dim parameter vector

$$\boxed{\mathbf{F(x)} = \arg\min_{y \in Y} w \cdot \phi(x, y)}$$

Key challenge: “Argmin” Inference

$$\mathrm{F(x)} = \arg\min_{y \in Y} w \cdot \phi(x, y)$$

Exponential size of output space !!

Key challenge: “Argmin” Inference

$$\mathbf{F(x)} = \arg\min_{y \in Y} w \cdot \phi(x, y)$$

- Time complexity of inference depends on the dependency structure of features $\phi(x, y)$

Key challenge: “Argmin” Inference

$$\mathrm{F(x)} = \arg\min_{y \in Y} w \cdot \phi(x, y)$$

- Time complexity of inference depends on the dependency structure of features $\phi(x, y)$
  - NP-Hard in general
  - Efficient inference algorithms exist only for simple features

Cost Function Learning: Generic Template

- **Training goal:**
  - Find weights w s.t
  - For each input x, the cost of the correct structured output y is lower than all wrong structured outputs

Exponential size of output space !!

- **repeat**
  - For every training example (x, y)
  - **Inference:** $\hat{y} = \arg min_{y \in Y} \, w \cdot \varphi(x, y)$
  - If mistake $y \neq \hat{y}$,

    Learning: online or batch weight update
- **until** *convergence* or *max. iterations*

Amortized Inference and Learning: Motivation

- We need to solve many inference problems during both training and testing
  - Computationally expensive

- Can we improve the speed of solving new inference problems based on past problem-solving experience?
  - Yes, amortized Inference!
  - Highly related to ``speedup learning'' [Fern, 2010]

Amortized Inference and Learning: Generic Approach

- Abstract out inference solver as a computational search process
- Learn search-control knowledge to improve the efficiency of search

- Example #1: ILP inference as branch-and-bound search and learn heuristics/policies
- Example #2: Learn search control knowledge for randomized greedy search based inference (Our focus)

Inference Solver: Randomized Greedy Search (RGS)

- Start from a random structured output
- Perform greedy search guided by scoring function $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y})$
- Stop after reaching local optima: y_{local}

- Accuracy of inference depends critically on the starting structured outputs

$\Downarrow$

- **Solution:** Multiple restarts and select the best local optima

Inference Solver: RGS

Repeat R_{max} times

- Start from a random structured output
- Perform greedy search guided by scoring function $\boldsymbol{F(x, y)}$
- Stop after reaching local optima: y_{local}

Prediction $\hat{y}$: best local optima

- Potential drawbacks
  - Requires large number of restarts to achieve high accuracy
  - May not work well for large outputs (# of output variables)

Inference Solver: RGS(α)

Repeat R_{max} times

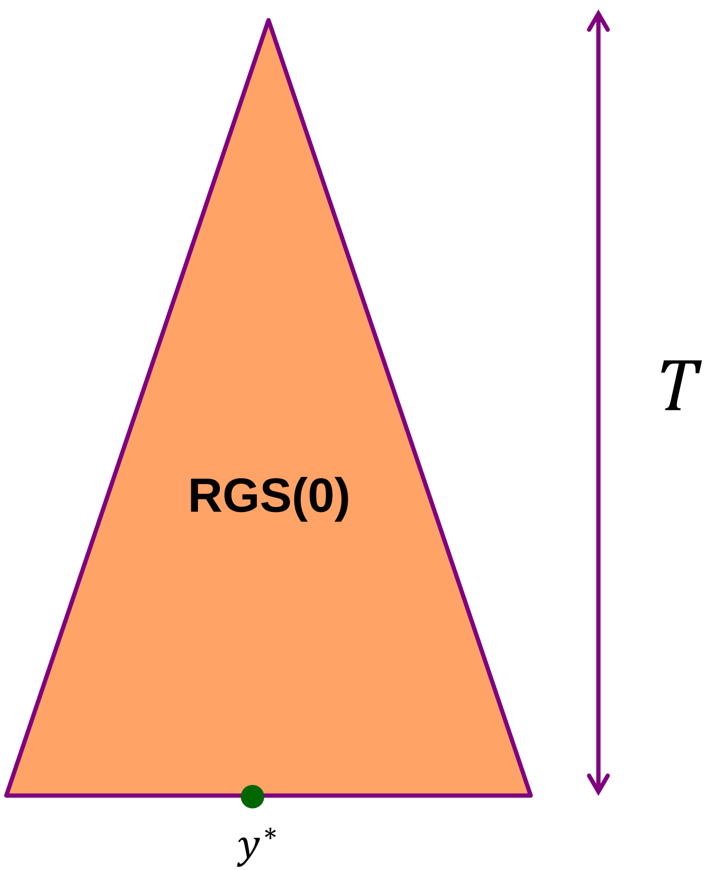
- α fraction of the output variables are initialized with a learned IID classifier
- Perform greedy search guided by scoring function $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y})$
- Stop after reaching local optima: y_{local}

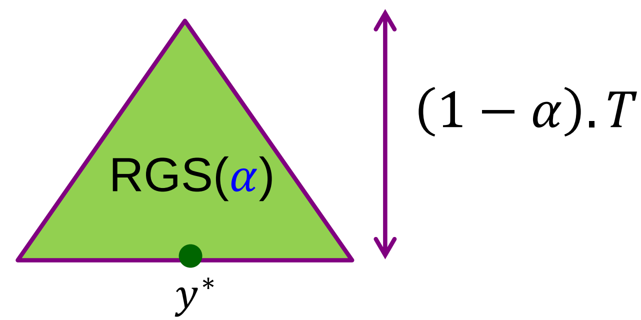
Prediction $\hat{y}$: best local optima

- RGS(0) is a special case [Zhang et al., 2014; Zhang et al., 2015]
  - ALL output variables are initialized randomly

Inference Solver: RGS(α)

- α controls the trade-off between
  - diversity of starting outputs
  - the minimum depth at which target outputs can be located





- Large α $\longrightarrow$ small target depth
- Can help for tasks with large outputs (e.g., coreference resolution)

Amortized RGS Inference: The Problem

- Given a set of structured inputs $\boldsymbol{D_x}$ and scoring function $\boldsymbol{F(x, y)}$ to score candidate outputs

- Reduce the number of iterations of RGS(α) to uncover high-scoring structured outputs

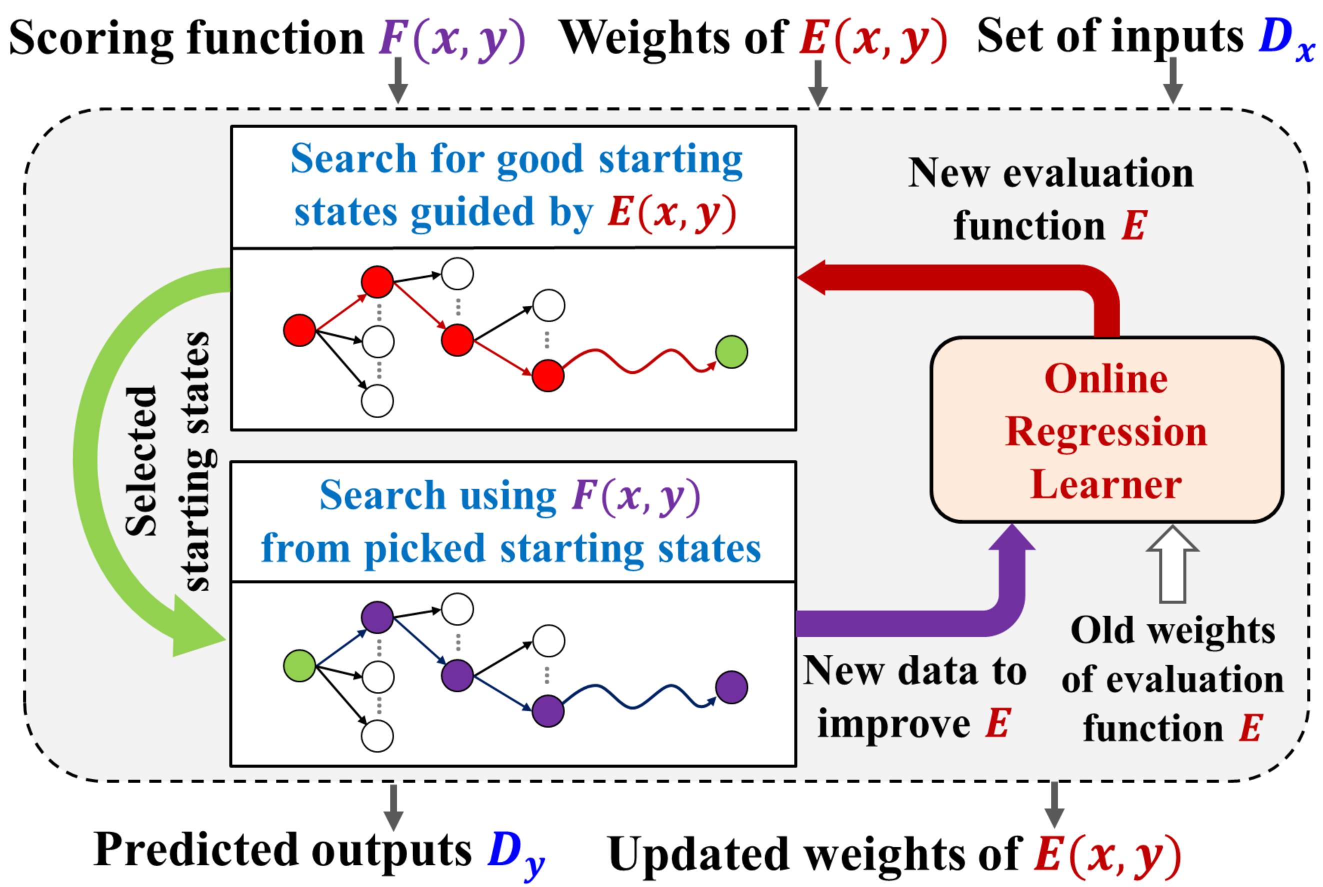
Amortized RGS Inference: Solution

- Given a set of structured inputs $\boldsymbol{D_x}$ and scoring function $\boldsymbol{F(x,y)}$ to score candidate outputs
- Reduce the number of iterations of RGS(α) to uncover high-scoring structured outputs

- Learn search control knowledge to select good starting states [Boyan and Moore, 2000]

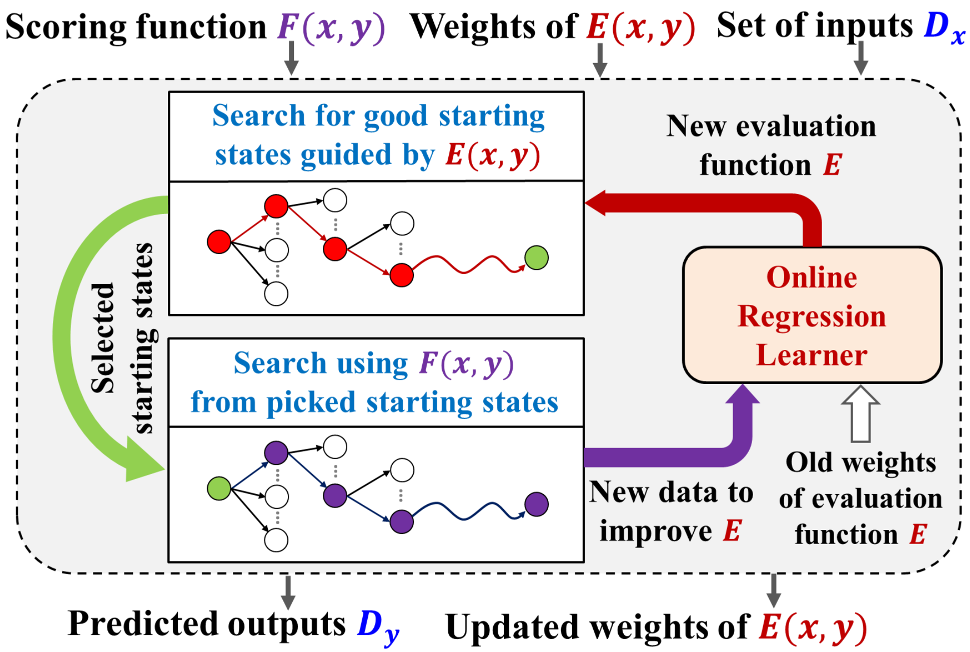
Amortized RGS Inference: Solution

- **Key Idea:** Learn evaluation function $E(x,y)$ to select good starting states to improve the accuracy of greedy search guided by $F(x,y)$ [Boyan and Moore, 2000]



Structured Learning w/ Amortized RGS

- Plug amortized RGS inference solver in the inner loop for learning weights of scoring function $F(x, y)$



$E(x, y)$ adapts to the changes in $F(x, y)$

Benchmark Domains

- **Sequence Labeling**
  - Handwriting recognition (*HW-Small* and *HW-Large*) [Taskar et al., 2003]
  - NET-Talk (*Stress* and *Phoneme* prediction) [Sejnowski and Rosenberg, 1987]
  - *Protein* secondary structure prediction [Dietterich et al., 2008]
  - *Twitter POS tagging* [Tu and Gimpel, 2008]
- **Multi-Label Classification**
  - 3 datasets: *Yeast*, *Bibtex*, and Bookmarks
- **Coreference Resolution**
  - *ACE2005* dataset (~ 50 to 300 mentions) [Durrett and Klein, 2014]
- **Semantic Segmentation of Images**
  - *MSRC* dataset (~ 700 super-pixels per image)

Evaluation Metrics: Task Loss Functions

- **Sequence Labeling**
  - Hamming accuracy
- **Multi-Label Classification**
  - Hamming accuracy, Example-F1, Example accuracy
- **Coreference Resolution**
  - MUC, B-Cube, CEAF, and CNNL Score
- **Image segmentation**
  - Pixel-wise classification accuracy

Baseline Methods

- Conditional Random Fields (CRFs)
- SEARN
- Cascades
- HC-Search
- Bi-LSTM (w./w.o. CRFs)
- Seq2Seq with Beam Search Optimization
- Structured SVM w/ RGS(0) inference with 50 restarts
- Structured SVM w/ RGS(α) inference

RGS(0) vs. RGS(α)

a. Sequence Labeling						
	HW-Small	HW-Large	Phoneme	Stress	TwitterPos	Protein
RGS(0)	92.32	97.83	82.28	80.84	89.9	62.75
RGS(α)	92.56	97.96	82.45	81.00	90.2	65.20

b. Multi-label Classification									
	Yeast			Bibtex			Bookmarks		
	Hamming	*ExmpF1*	*ExmpAcc*	*Hamming*	*ExmpF1*	*ExmpAcc*	*Hamming*	*ExmpF1*	*ExmpAcc*
RGS(0)	80.04	63.90	52.18	98.12	44.11	36.65	99.13	36.88	31.46
RGS(α)	80.10	63.90	52.90	98.62	44.86	36.78	99.15	36.98	31.58

c. Coreference Resolution (ACE 2005)				
	MUC	*BCube*	*CEAFe*	*CoNLL*
RGS(0)	80.07	74.13	71.25	75.15
RGS(α)	82.18	76.57	74.01	77.58

d. Image Segmentation (MSRC)		
	Global	*Average*
RGS(0)	81.27	73.14
RGS(α)	85.29	78.92

Algorithms Datasets *Metrics*

✓ RGS with best α gives better accuracy than RGS(0) for tasks with large structured outputs.

RGS(α) vs. State-of-the-art

a. Sequence Labeling						
	HW-Small	HW-Large	Phoneme	Stress	TwitterPos	Protein
Cascades	89.18	97.84	82.59	80.49	-	-
HC-Search	89.96	97.79	85.71	83.68	-	-
CRF	80.03	86.89	78.91	78.52	-	62.44
SEARN	82.12	90.58	77.26	76.15	-	-
BiLSTM	83.18	92.50	77.98	76.55	88.8	61.26
BiLSTM-CRF	88.78	95.76	81.03	80.14	89.2	62.79
Seq2Seq(Beam=1)	83.38	93.65	78.82	79.62	89.1	62.90
Seq2Seq(Beam=20)	89.38	98.95	82.31	81.5	90.2	63.81
RGS(α)	92.56	97.96	82.45	81.00	90.2	65.20

- ✓ RGS(α) is competitive or better than many state-of-the-art methods.

Note**: ***BiLSTM-CRF is the CRF model with BiLSTM hidden states as unary features.

RGS(α) vs. State-of-the-art

b. Multi-label Classification									
	Yeast			Bibtex			Bookmarks		
	Hamming	*ExmpF1*	*ExmpAcc*	*Hamming*	*ExmpF1*	*ExmpAcc*	*Hamming*	*ExmpF1*	*ExmpAcc*
SPEN(E2E)	79.6	63.8	52.0	98.5	42.1	36.8	99.1	35.6	29.3
DVN	78.9	63.8	51.9	98.5	44.7	37.2	99.1	37.1	30.1
InfNet	79.4	63.6	51.7	98.1	42.2	37.1	99.2	37.6	30.9
RGS(α)	80.10	63.90	52.90	98.62	44.86	36.78	99.15	36.98	31.58

c. Coreference Resolution (ACE 2005)				
	MUC	*BCube*	*CEAFe*	*CoNLL*
Berkeley	81.41	74.70	72.93	76.35
RGS(α)	82.18	76.57	74.01	77.58

d. Image Segmentation (MSRC)		
	Global	*Average*
ICCV2011	85	77
CRF-CNN	91.1	90.5
RGS(α)	85.29	78.92
RGS(α)-CNN*	91.53	90.28

✓ RGS(α) is competitive or better than many state-of-the-art methods.

Note**: ***RGS(α)-CNN *is the RGS(α) with* 7th layer AlexNet output as unary features.

Test-time Inference: Amortized RGS vs. RGS

Testing Time (*milli seconds*)					
	Bibtex	Bookmarks	HWLarge	MSRC	ACE05
SPEN(E2E)	5791	63073	-	-	-
DVN	18086	211448	-	-	-
RGS	69890	288058	17323	9451	282864
A-RGS	20925	98921	4812	2294	55355

✓ A-RGS is 3 to 5 times faster than the RGS approach.

✓ The speedup factor for A-RGS is higher for tasks with large structured outputs.

Training Results: Amortized RGS vs. RGS

Training Time (*minutes*)						
	Yeast	Bibtex	Bookmarks	HWLarge	MSRC	ACE05
SPEN(E2E)	19	114	237	-	-	-
DVN	4	20	204	-	-	-
DCD(RGS)	9	95	392	71	115	171
DCD(A-RGS)	5	32	319	44	27	39

- ✓ DCD(A-RGS) training takes shorter time than DCD(A-RGS)
- ✓ DVN and SPEN(E2E) perform better on Bibtex and Bookmarks, whereas DCD(A-RGS) perform comparably or better on Yeast. Most time (~89%) was consumed on updating the dual weights.

Questions ?