Daumé III, Langford, Marcu: Search-based Structured Prediction

SEARN

Julia Kreutzer

Seminar „Algorithms for Learning and Search in Structured Prediction“
Institut für Computerlinguistik
Universität Heidelberg

May 12, 2015

Overview

1. Introduction
2. The SEARN algorithm
3. SEARN analysed
4. SEARN in experiments
5. Conclusion

Introduction

Recap: What is structured prediction? Why/where it is challenging?

Structured Prediction

- Structured Prediction Problem

  $(x, \boldsymbol{c}) \sim D$ with inputs $x \in X$, cost vectors $c \in (\mathbb{R}^+)^k$, k labels

- Goal

  Find $h: X \rightarrow Y$ that minimizes $L(D, h) = \mathbb{E}_{(x,\boldsymbol{c}) \sim D} \{c_h(x)\}$

Structured Prediction

- Structured Prediction Problem

  $(x, \boldsymbol{c}) \sim D$ with inputs $x \in X$, cost vectors $c \in (\mathbb{R}^+)^k$, k labels
- Goal

  Find $h: X \rightarrow Y$ that minimizes $L(D, h) = \mathbb{E}_{(x,\boldsymbol{c}) \sim D} \{c_h(x)\}$
- Challenges

  Exact search is not always tractable

  Loss functions are not decomposable

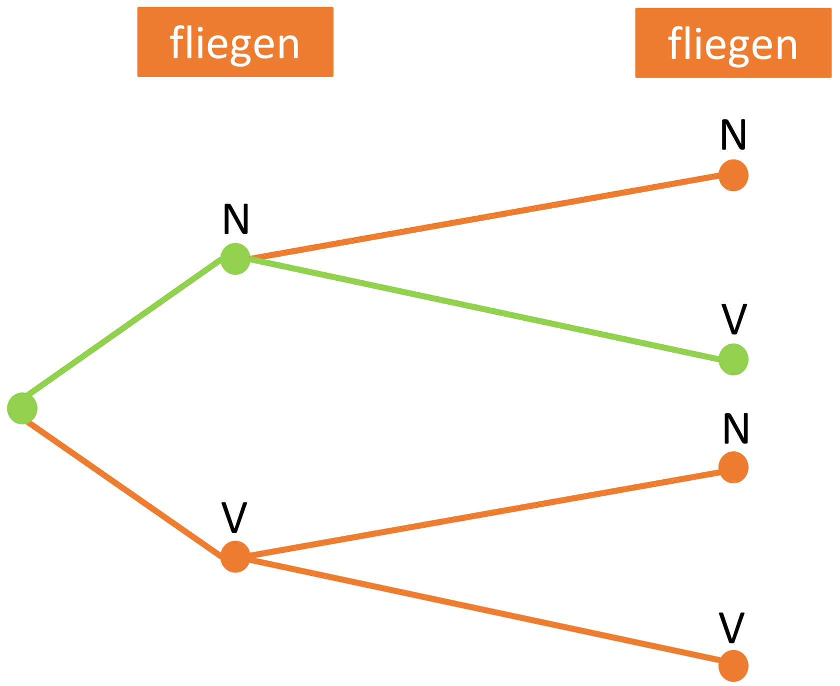
  Complex feature functions

Structured Prediction

- Structured Prediction Problem
  $(x, \boldsymbol{c}) \sim D$ with inputs $x \in X$, cost vectors $c \in (\mathbb{R}^{+})^{k}$, k labels
- Goal
  Find $h: X \rightarrow Y$ that minimizes $L(D, h) = \mathbb{E}_{(x,\boldsymbol{c}) \sim D} \{c_h(x)\}$
- Challenges
  Exact search is not always tractable
  Loss functions are not decomposable
  Complex feature functions
- [5] Approximate search instead of exact search: „enqueue“
- [6] Under-generating vs. over-generating algorithms

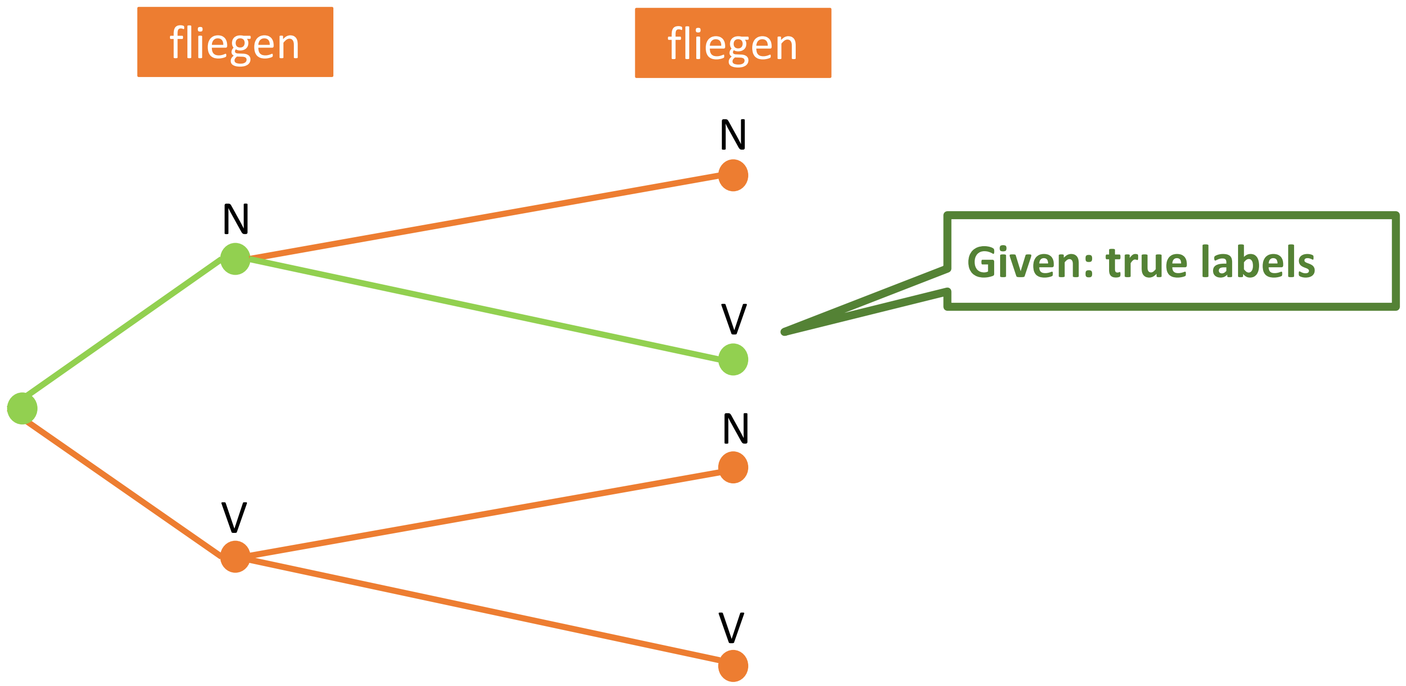
Motivating Example

- POS-Tagging
  - Simplest setting: two words, two possible labels



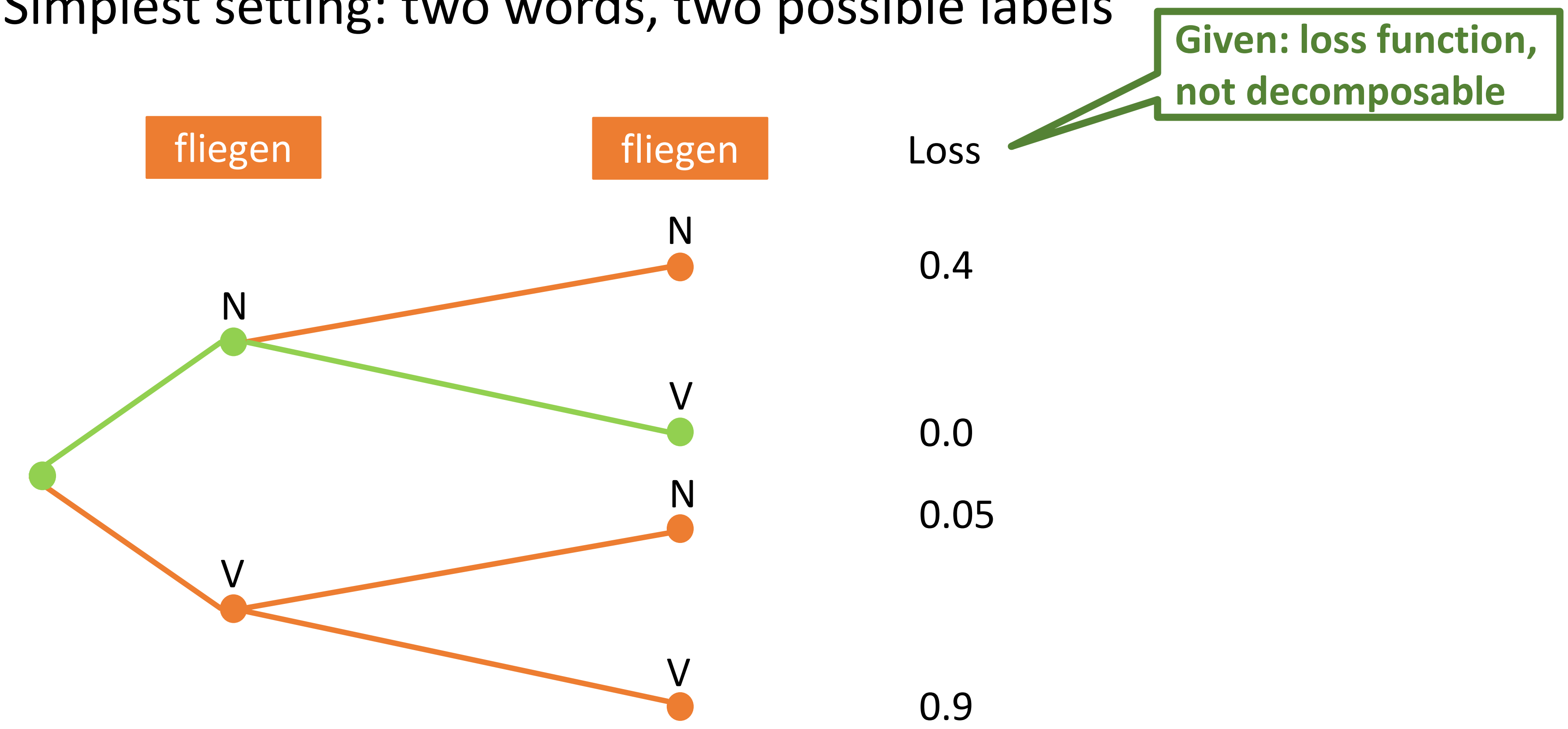
Motivating Example

- POS-Tagging
  - Simplest setting: two words, two possible labels



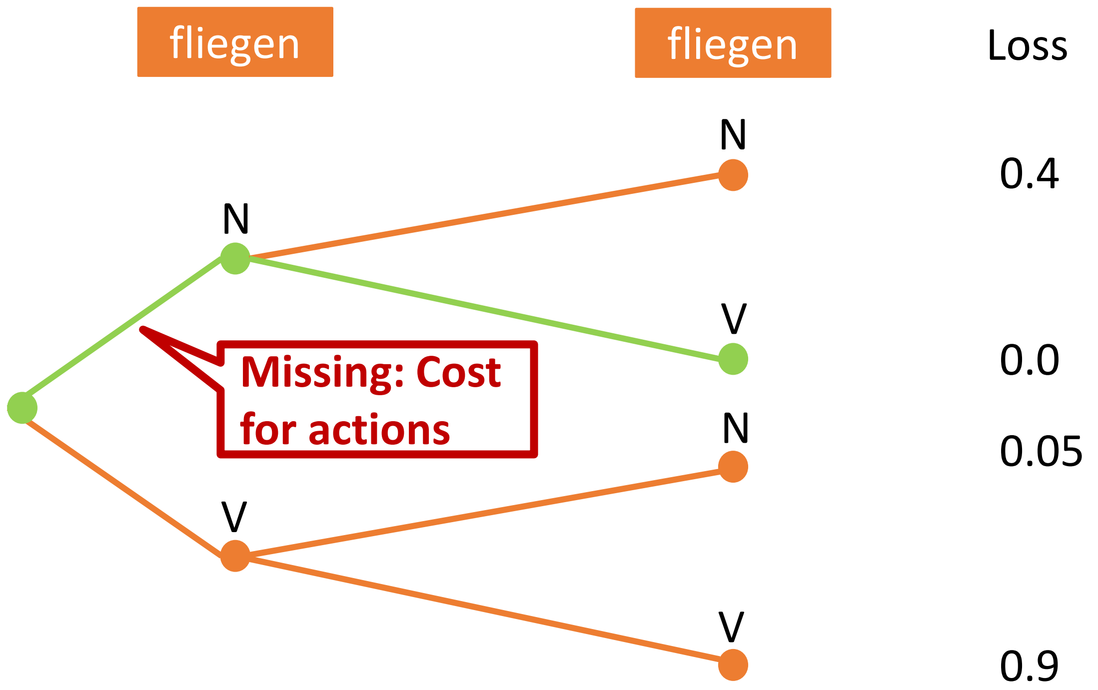
Motivating Example

- POS-Tagging
  - Simplest setting: two words, two possible labels



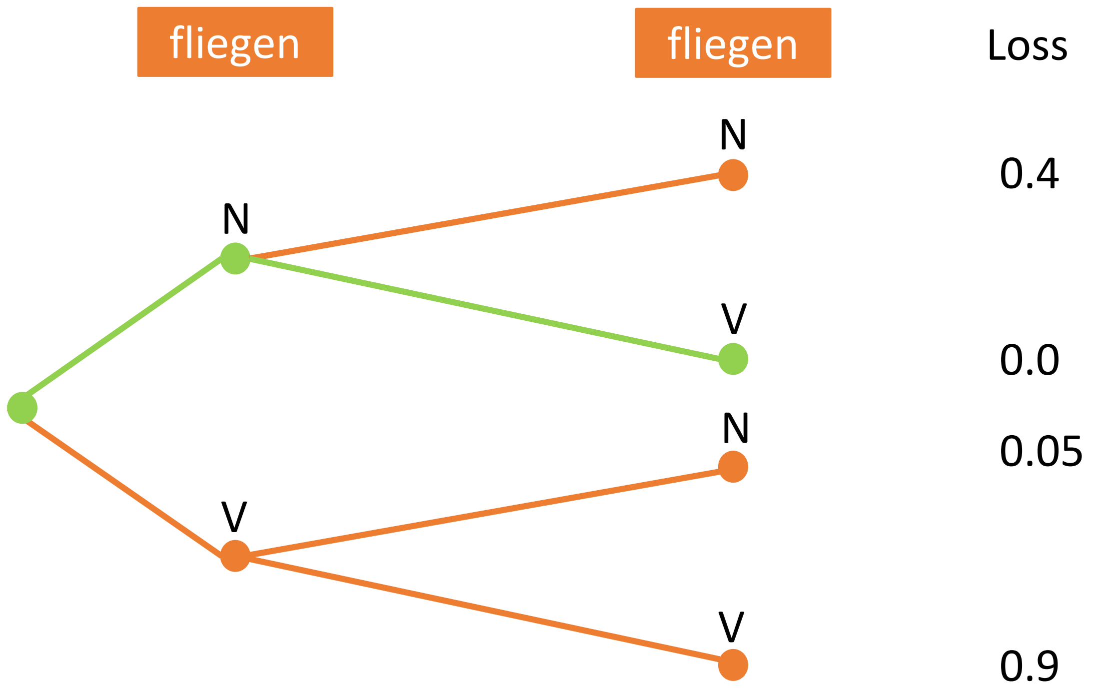
Motivating Example

- POS-Tagging
  - Simplest setting: two words, two possible labels



Motivating Example

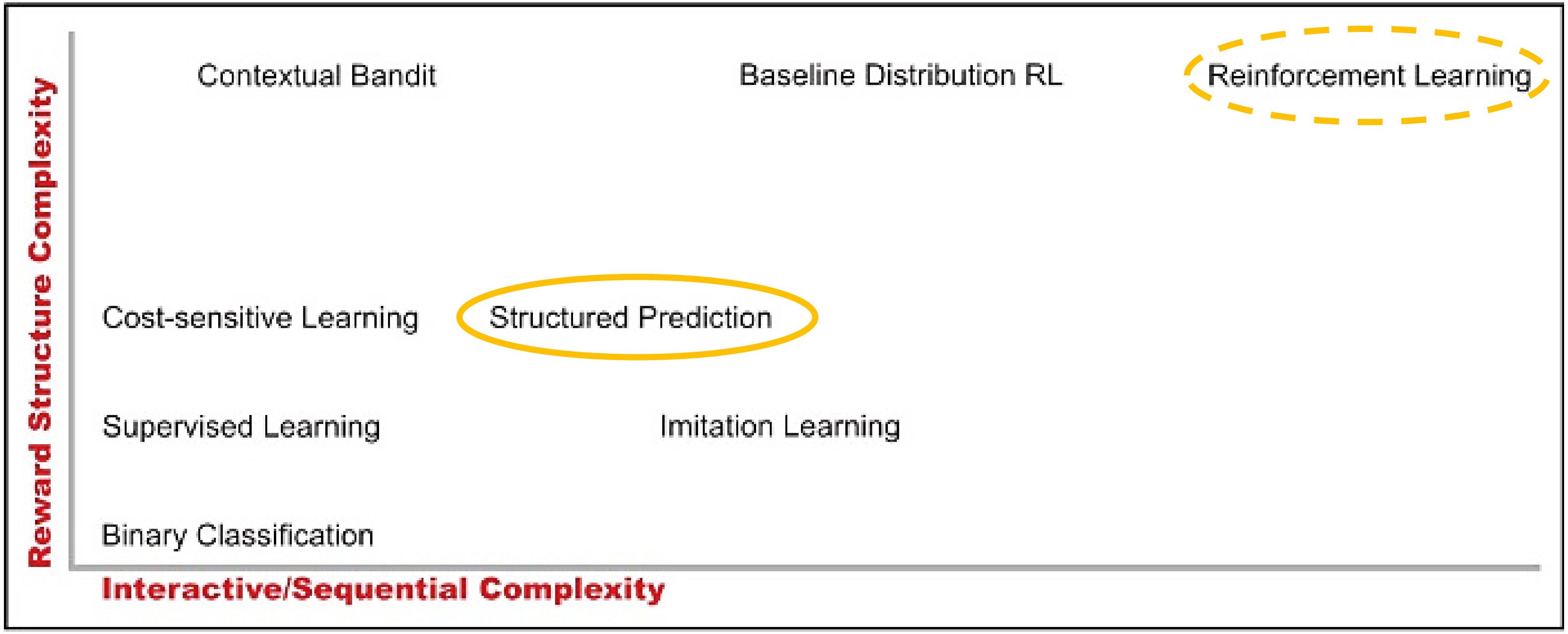
- POS-Tagging
  - Simplest setting: two words, two possible labels



The SEARN algorithm

What kind of algorithm is it? Which problem does it solve? What is special?

Structured Prediction in Context



Source: [7]

Characteristics

- SEARN:
  - Meta-algorithm

    How can we learn from a teacher?
  - **Sear**ch + **Learn**

    View the problem as a search problem

    Learn a classifier that walks through search space in a good way

    Instead of training on true path: train on path that is actually taken in practice
  - Any loss function
  - Any class of features

Components

- A search space S
- A cost-sensitive learning algorithm A
- Training data: structured, labeled
- A loss function $L(\boldsymbol{y}, f(\widehat{\boldsymbol{y}}))$
- A good initial policy $\pi(s, \boldsymbol{c})$

Components

- A search space S
  - State in the search space: $s = x \times (y_1, \ldots, y_T)$
  - Final elements: sequence of choices $\hat{\boldsymbol{y}}$
  - Abstract: $\mathrm{f}(\hat{\boldsymbol{y}})$
  - Concrete: $\mathrm{f}(\hat{\boldsymbol{y}}) = \hat{\boldsymbol{y}}$

Components

- A search space S
- A cost-sensitive learning algorithm A

    Multiclass classifier $h(s)$ for location in search space s

    Trained on cost-sensitive training data

    „policy“ (→ reinforcement learning)

Components

- A search space S
- A cost-sensitive learning algorithm A
- Training data: structured, labeled

  $(\boldsymbol{x}, \boldsymbol{y}) \in S^{SP}$

  $\boldsymbol{y} \in Y$ decompose into vectors $(y_0, y_1, \dots, y_T)$

  Arbitrary set of labels

Components

- A search space S
- A cost-sensitive learning algorithm A
- Training data: structured, labeled
- A loss function $L(\boldsymbol{y}, f(\widehat{\boldsymbol{y}}))$

  Computable for any full-length prediction sequence $(y_0, y_1, \ldots, y_T)$

  Does not have to be decomposable

Components

- A search space S
- A cost-sensitive learning algorithm A
- Training data: structured, labeled
- A loss function $L(\boldsymbol{y}, f(\widehat{\boldsymbol{y}}))$
- A good initial policy $\pi(s, \boldsymbol{c})$

  Achieves low loss on training data

  „the teacher“

Prediction

- At test time:
  - Use returned policy
  - Compute y_0 on basis of x

    Compute y_1 on basis of y_0 and x

    ...

    Compute y_T on basis of $x, y_0, y_1, \ldots y_{T-1}$

Prediction

- At test time:
  - Use returned policy
  - Compute y_0 on basis of x

    Compute y_1 on basis of y_0 and x

    …

    Compute y_T on basis of $x, y_0, y_1, \ldots y_{T-1}$
- No Markov assumption
- Feature function is essential

Training

Algorithm SEARN(S^{SP}, π, A)

Initialize policy $h \leftarrow \pi$

while h has a significant dependence on π **do**

  Initialize the set of cost-sensitive examples $S \leftarrow \emptyset$

  for $(x, y) \in S^{SP}$ **do**

    Compute predictions under the current policy $\hat{\boldsymbol{y}} \sim x, h$

    for $t = 1 \ldots T_x$ **do**

      Compute features $\Phi = \Phi(s_t)$ for state $s_t = (x, y_1, ..., y_t)$

      Initialize a cost vector $\boldsymbol{c} = \langle\rangle$

      for each possible action a **do**

        Let the cost ℓ_a for example $x, \mathbf{c}$ at state s be $\ell_h(\mathbf{c}, s, a)$

      end for

      Add cost-sensitive example $(\Phi, \boldsymbol{\ell})$ to S

    end for

  end for

  Learn a classifier on S: $h' \leftarrow A(S)$

  Interpolate: $h \leftarrow \beta h' + (1 - \beta) h$

end while

return h_{last} without π

Initialization with optimal policy

Convergence criterion

Generate path over training sample

Features for state at timestep t

Generate multiclass examples for possible decisions and losses based on current policy

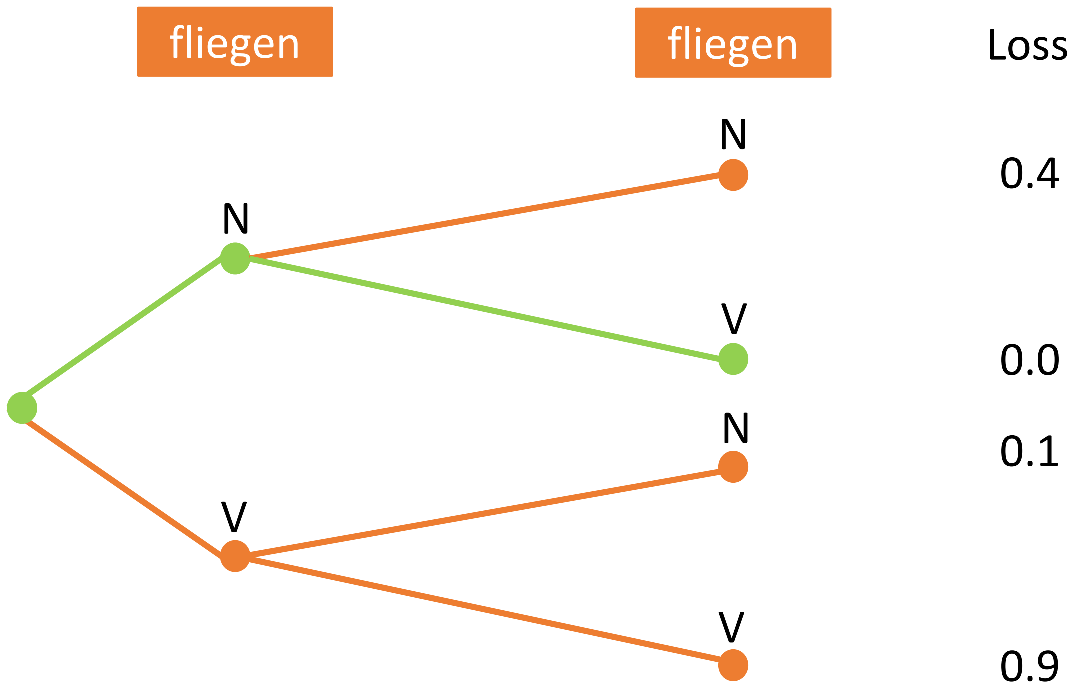
Train new classifier on examples

Combine old and new classifier

Source: [1]

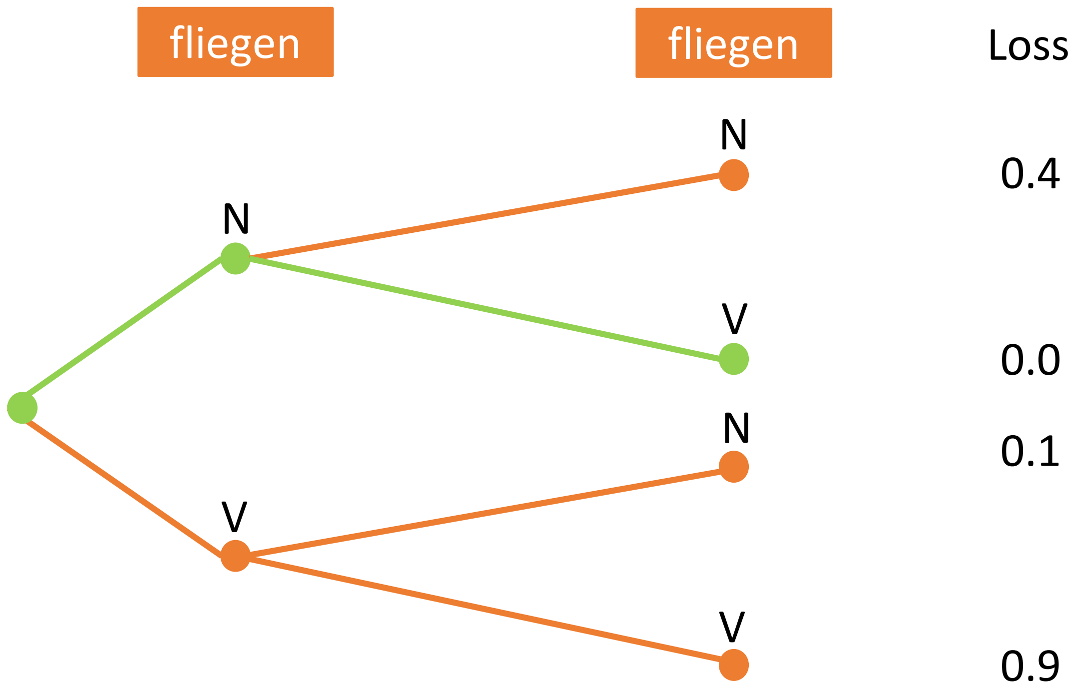
Motivating Example re-visited

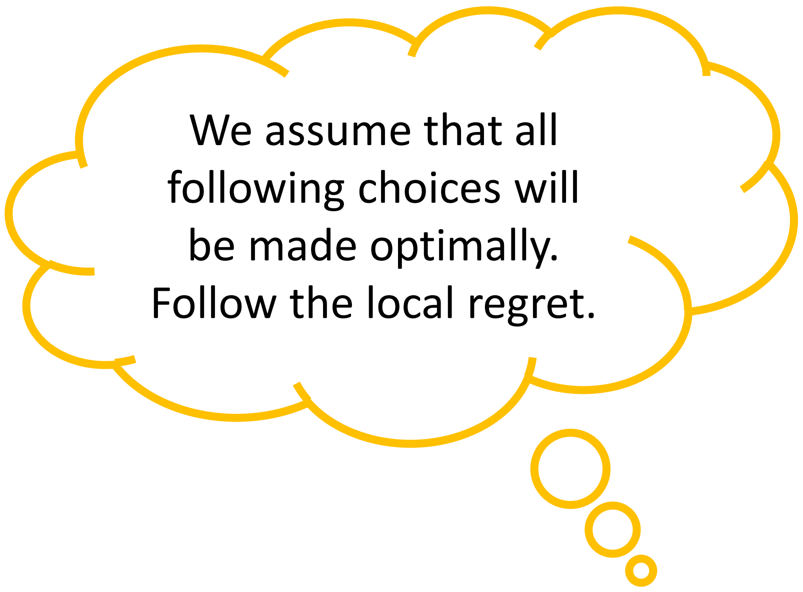
- POS-Tagging
  - Simplest setting: two words, two possible labels



Motivating Example re-visited

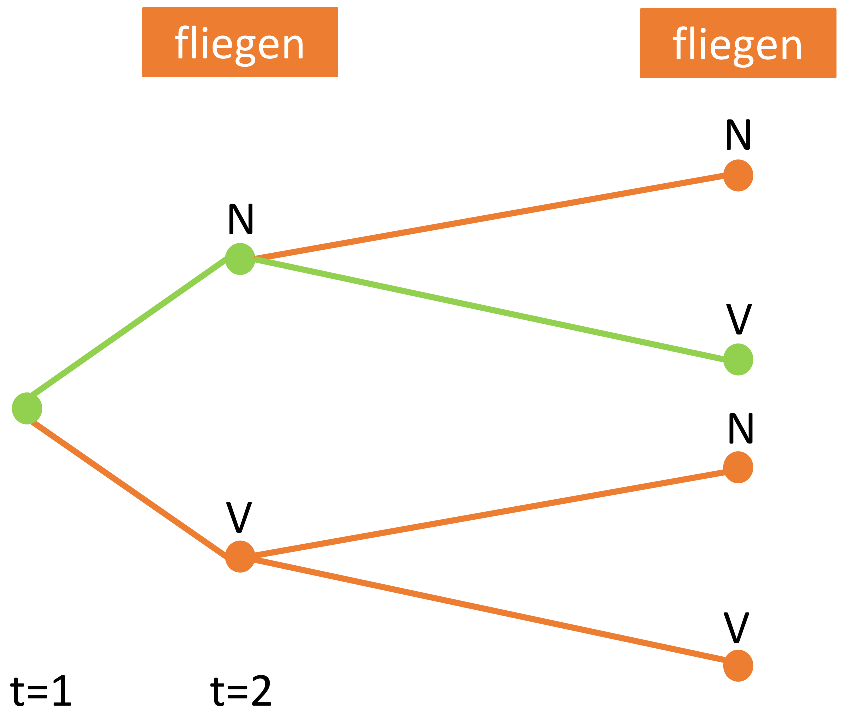
- POS-Tagging
  - Simplest setting: two words, two possible labels





Motivating Example re-visited

- POS-Tagging
  - Simplest setting: two words, two possible labels



Loss

0.4

0.0

0.1

0.9

Computing the regret

t=1

a=„N“ $l_a = 0.2$

a=„V“ $l_a = 0.5$

$\mathbf{l}_1 = \langle 0.2, 0.5 \rangle$

t=2

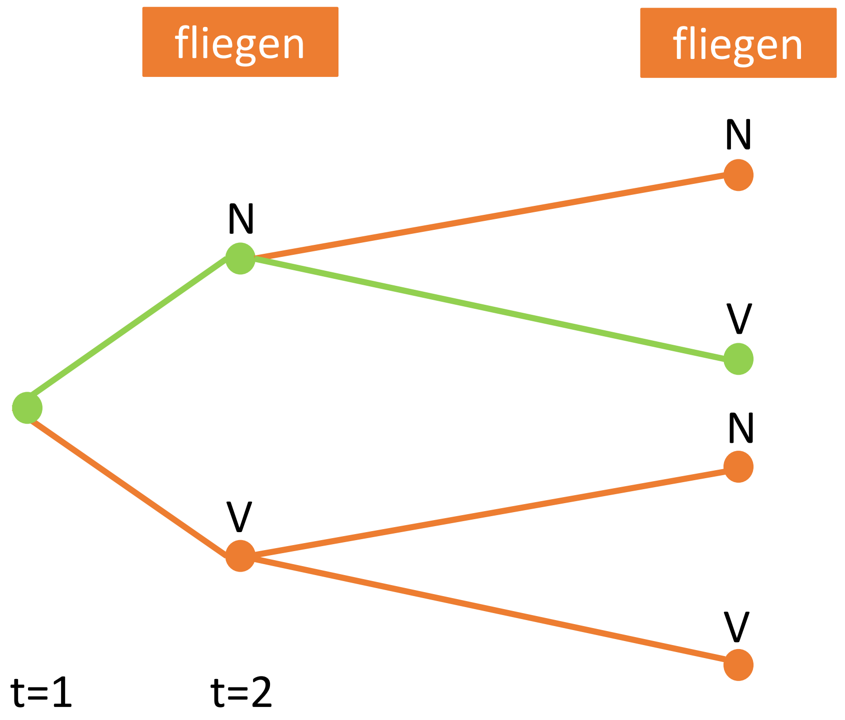
a=„N“ $l_a = 0.4$

a=„V“ $l_a = 0.0$

$\mathbf{l}_2 = \langle 0.0, 0.4 \rangle$

Motivating Example re-visited

- POS-Tagging
  - Simplest setting: two words, two possible labels



Loss
0.4
0.0
0.1
0.9

Creating cost-sensitive examples

$\mathbf{l}_1$ = <0.2, 0.5>

features for state: $\Phi_1 = \Phi(x)$

$\mathbf{l}_2$ = <0.0, 0.4>

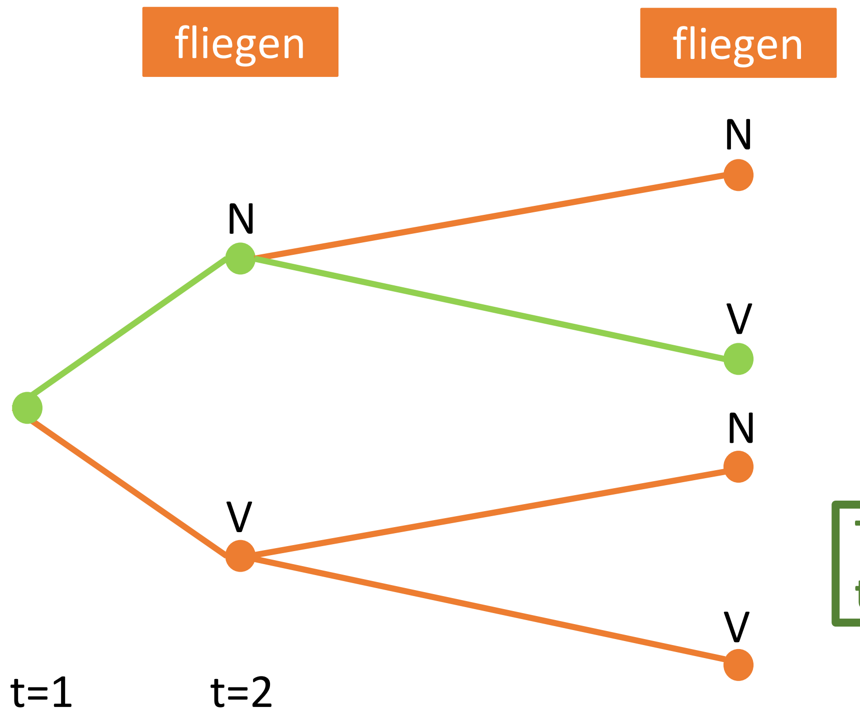
features for state: $\Phi_2 = \Phi(x, y_1)$

add to S:

$\{<\Phi_1, \mathbf{l}_1>, <\Phi_2, \mathbf{l}_2>\}$

Motivating Example re-visited

- POS-Tagging
  - Simplest setting: two words, two possible labels



Loss

0.4

0.0

0.1

0.9

Creating cost-sensitive examples

$\mathbf{l}_1 = <0.2, 0.5>$

features for state: $\Phi_1 = \Phi(x)$

$\mathbf{l}_2 = <0.0, 0.4>$

features for state: $\Phi_2 = \Phi(x, y_1)$

Train classifier on these tuples!

add to S:

$\{<\Phi_1, \mathbf{l}_1>, <\Phi_2, \mathbf{l}_2>\}$

Details

- Initial policy
  - Takes full advantage of training data labels
  - Use search to create the initial policy (if not available analytically):
    - Given a node in the search space with cost vector, compute the best step to take
    - = given a node in the search space, find the shortest way to a goal
  - Optimal approximation: assume all further decisions will be made optimally
  - „greedy" search
  - Choice of search algorithm influences bias in learning algorithm

$$\pi(s, \boldsymbol{c}) = \arg\min_{y_{t+1}} \min_{y_{t+2},\dots,y_T} c\langle y_1, \dots, y_T \rangle$$

Details

- Cost-sensitive examples
  - Run the given policy h over the training data
  - Prediction is sequence $\hat{\boldsymbol{y}}$ with loss $\boldsymbol{c}_{\hat{\boldsymbol{y}}}$
  - Compute (arbitrary) features $\varphi = \varphi(s)$ for state s on sequence
  - Compute cost („regret“) for each state s and each action a:

$$l_h(\boldsymbol{c}, s, a) = \mathbb{E}_{\hat{y}\sim(s,a,h)}\boldsymbol{c}_{\hat{\boldsymbol{y}}} - \min_{a'} \mathbb{E}_{\hat{y}\sim(s,a',h)}\boldsymbol{c}_{\hat{\boldsymbol{y}}}$$

→ $(\varphi, \boldsymbol{l}) \in S$ is the input data structure for the learner

SEARN analysed

What can we tell about SEARN from an analytical perspective?

SEARN analysed

- Moving from initial policy to fully learned policy
- Each iteration „degrades“ current policy
- Analysis shows: the degradation is small
  - Theorem 2: Upper bound on the loss of a learned classifier
  - Lemma 1: Upper bound on the loss of a classifier after first iteration
  - Lemma 2: Upper bound after several iterations

SEARN analysed

- Proof for lemma 1:
  - Interpolation

    $$h^{new} \leftarrow \beta h' + (1-\beta)h$$
  - Maximal cost:

    $$c_{max} = \mathbb{E}_{(x,\boldsymbol{c})\sim D} \max_i c_i$$
  - Cases:
    1. Learned policy is never called
    2. Called once
    3. Called more than once
  - Assumption:

    $$\beta < 1/T$$

$$\begin{aligned} L(\mathcal{D}, h^{\text{new}}) =& Pr(c=0)L(\mathcal{D}, h^{\text{new}} \mid c=0) \\ &+ Pr(c=1)L(\mathcal{D}, h^{\text{new}} \mid c=1) \\ &+ Pr(c\geq 2)L(\mathcal{D}, h^{\text{new}} \mid c\geq 2) \end{aligned} \tag{6}$$

$$\begin{aligned} \leq& (1-\beta)^T L(\mathcal{D}, h) + T\beta(1-\beta)^{T-1}\Big[L(\mathcal{D}, h) + \ell_h^{\text{CS}}(h')\Big] \\ &+ \Big[1-(1-\beta)^T - T\beta(1-\beta)^{T-1}\Big]c_{\max} \end{aligned} \tag{7}$$

$$\begin{aligned} =& L(\mathcal{D}, h) + T\beta(1-\beta)^{T-1}\ell_h^{\text{CS}}(h') + \left(\sum_{i=2}^{T}(-1)^i\beta^i\binom{T}{i}\right)L(\mathcal{D}, h) \\ &+ \Big[1-(1-\beta)^T - T\beta(1-\beta)^{T-1}\Big]c_{\max} \end{aligned} \tag{8}$$

Source: [1]

SEARN analysed

- Proof for lemma 1:
  - Interpolation

    $$h^{new} \leftarrow \beta h' + (1-\beta)h$$
  - Maximal cost:

    $$c_{max} = \mathbb{E}_{(x,\boldsymbol{c})\sim D} \max_i c_i$$
  - Cases:
    1. Learned policy is never called
    2. Called once
    3. Called more than once
  - Assumption:

    $$\beta < 1/T$$

$$\begin{aligned} &\leq L(\mathcal{D},h) + T\beta\ell_h^{\mathrm{CS}}(h') \\ &\quad + \left[1-(1-\beta)^T - T\beta(1-\beta)^{T-1}\right](c_{\max} - L(\mathcal{D},h)) \end{aligned} \tag{9}$$

$$\begin{aligned} &\leq L(\mathcal{D},h) + T\beta\ell_h^{\mathrm{CS}}(h') \\ &\quad + \left[1-(1-\beta)^T - T\beta(1-\beta)^{T-1}\right]c_{\max} \end{aligned} \tag{10}$$

$$= L(\mathcal{D},h) + T\beta\ell_h^{\mathrm{CS}}(h') + \left(\sum_{i=2}^{T}(-1)^i\beta^i\binom{T}{i}\right)c_{\max} \tag{11}$$

$$\leq L(\mathcal{D},h) + T\beta\ell_h^{\mathrm{CS}}(h') + \frac{1}{2}T^2\beta^2 c_{\max} \tag{12}$$

Source: [1]

SEARN in comparison

Why/where is SEARN superior to other structured prediction algorithms?

SEARN in comparison

- Independent models
  - Ignore structure or constrain membership

    No complex features, limited to Hamming loss

    = SEARN with features independent of history
  - Maximum Entropy Markov Model (MEMM)

    Prediction on basis of k previous predictions

    Assumption: previous predictions are correct → can perform arbitrarily bad

    Stacked MEMM‘s: SEARN with $\beta = 1$ limited to sequence labeling

SEARN in comparison

- Perceptron-based models
  - Structured Perceptron

    Assumption: argmax is tractable

    SEARN in reverse: moving from incorrect towards true output
  - Incremental Perceptron

    Replace argmax with beam search

    Limitations: beam-search applications, decomposable loss function

- Global models
  - Conditional Random Field & Max-Margin Markov Network (M^3N)

    In application limited to linear chain models with Markov assumption

    SEARN is more general

Experiments

How can we apply SEARN to structured prediction tasks? Does it perform well?

Experiments

1. Sequence Labeling
    - Handwriting recognition
    - Spanish NER
    - Syntactic chunking
    - Joint chunking and POS tagging

SEARN:

- Loss per label: Hamming loss
- Loss per chunk: F1
- Left-to-right greedy search
- Chunk-at-a-time decoding (BIO)
- Reduction to binary classification

$$l^{Ham}(y, \hat{y}) \triangleq \sum_{n=1}^{N} 1[y_n \neq \hat{y}_n]$$

$$l^{F}(y, \hat{y}) \triangleq \frac{2\,|y \cap \hat{y}|}{|y|+|\hat{y}|}$$

Experiments

1. Sequence Labeling

- Results

ALGORITHM	Handwriting Small	Handwriting Large	NER Small	NER Large	Chunk	C+T
CLASSIFICATION						
Perceptron	65.56	70.05	91.11	94.37	83.12	87.88
Log Reg	68.65	72.10	93.62	96.09	85.40	90.39
SVM-Lin	75.75	82.42	93.74	97.31	86.09	93.94
SVM-Quad	82.63	82.52	85.49	85.49	∼	∼
STRUCTURED						
Str. Perc.	69.74	74.12	93.18	95.32	92.44	93.12
CRF	–	–	94.94	∼	94.77	96.48
SVM^{struct}	–	–	94.90	∼	–	–
M^{3}N-Lin	81.00	∼	–	–	–	–
M^{3}N-Quad	87.00	∼	–	–	–	–
SEARN						
Perceptron	70.17	76.88	95.01	97.67	94.36	96.81
Log Reg	73.81	79.28	95.90	98.17	94.47	96.95
SVM-Lin	82.12	90.58	95.91	98.11	94.44	96.98
SVM-Quad	87.55	90.91	89.31	90.01	∼	∼

∼: could not scale

−: not reported

F1 on Chunk, C+T

Hamming on Handwriting, NER

Source: [1]

Experiments

2. Automatic Document Summarization

- Greedily extract sentences of a document until word limit reached
- Vine-growth model on syntactic dependency parse tree
- Actions: add root of new tree or child of already added node
- Loss: Rouge
- Initial Policy: argmax intractable (constraints), beam search approximation

Experiments

2. Automatic Document Summarization

- Results

Rouge score

	ORACLE		SEARN		BAYESUM			
	Vine	Extr	Vine	Extr	D05	D03	Base	Best
100 w	.0729	.0362	.0415	.0345	.0340	.0316	.0181	-
250 w	.1351	.0809	.0824	.0767	.0762	.0698	.0403	.0725

Source: [1]

Conclusion

What did we learn about SEARN? What did we not learn?

Summary

- Core idea: combining search and learning

  „Instead of accounting for search in the process of learning, I treat the structured prediction problem as being defined by a search process." [2]

- Meta-algorithm for structured prediction
  - Minimal requirements for structure and loss function
  - Start from good initial policy and generalize

- Competitive results for sequence labeling and summarization task

Problems and Questions

- The algorithm
  - Heavily relies on quality of initial policy
    - Efficiency
    - Bias
    - Noise
  - Definition of convergence criterion?
  - Missing details for SEARN in test time
  - Policy might be stochastic

Problems and Questions

- The application
  - Documented experiments lack interesting details
    - Iteration numbers
    - Observed speed of convergence
    - Interpolation and storage of classifiers
    - Use, integration and parametrization of base classifiers
  - Only a few experiments (by the same person)
  - (Un-)popularity in practice?
  - Machine Translation?

Problems and Questions

Your opinion! ☺

Problems and Questions

Your opinion! ☺

Thank you!

References

[1] Hal Daumé III, John Langford, and Daniel Marcu, **Search-Based Structured Prediction,** Machine Learning 75, no. 3 (June 2009): 297–325.

[2] Hal Daumé III, **Practical Structured Learning Techniques for Natural Language Processing** (ProQuest, 2006), PhD thesis at the University of Southern California.

[3] Hal Daumé III, John Langford, and Daniel Marcu, **Searn in Practice,** Unpublished, 2006.

[4] **Searn website**: http://www.umiacs.umd.edu/~hal/searn/ (last visited: May 10, 2015)

[5] Hal Daumé III and Daniel Marcu, **Learning as Search Optimization: Approximate Large Margin Methods for Structured Prediction,** in Proceedings of the 22nd International Conference on Machine Learning (ACM, 2005), 169–76.

[6] Thomas Finley and Thorsten Joachims, **Training Structural SVMs When Exact Inference Is Intractable,** in Proceedings of the 25th International Conference on Machine Learning (ACM, 2008), 304–11.

[7] J. Kober, J. A. Bagnell, and J. Peters, **Reinforcement Learning in Robotics: A Survey**, The International Journal of Robotics Research 32, no. 11 (September 1, 2013).