TTIC 31210:
Advanced Natural Language Processing

Kevin Gimpel

Spring 2019

Lecture 10:

Inference & Learning in Structured Prediction

Roadmap

- intro (1 lecture)
- deep learning for NLP (5 lectures)
- **structured prediction (4 lectures)**
  - introducing/formalizing structured prediction, categories of structures
  - **inference: dynamic programming, greedy algorithms, beam search**
  - **inference with non-local features**
  - **learning in structured prediction**
- generative models, latent variables, unsupervised learning, variational autoencoders (2 lectures)
- Bayesian methods in NLP (2 lectures)
- Bayesian nonparametrics in NLP (2 lectures)
- review & other topics (1 lecture)

Assignments

- Assignment 2 due today. Questions?

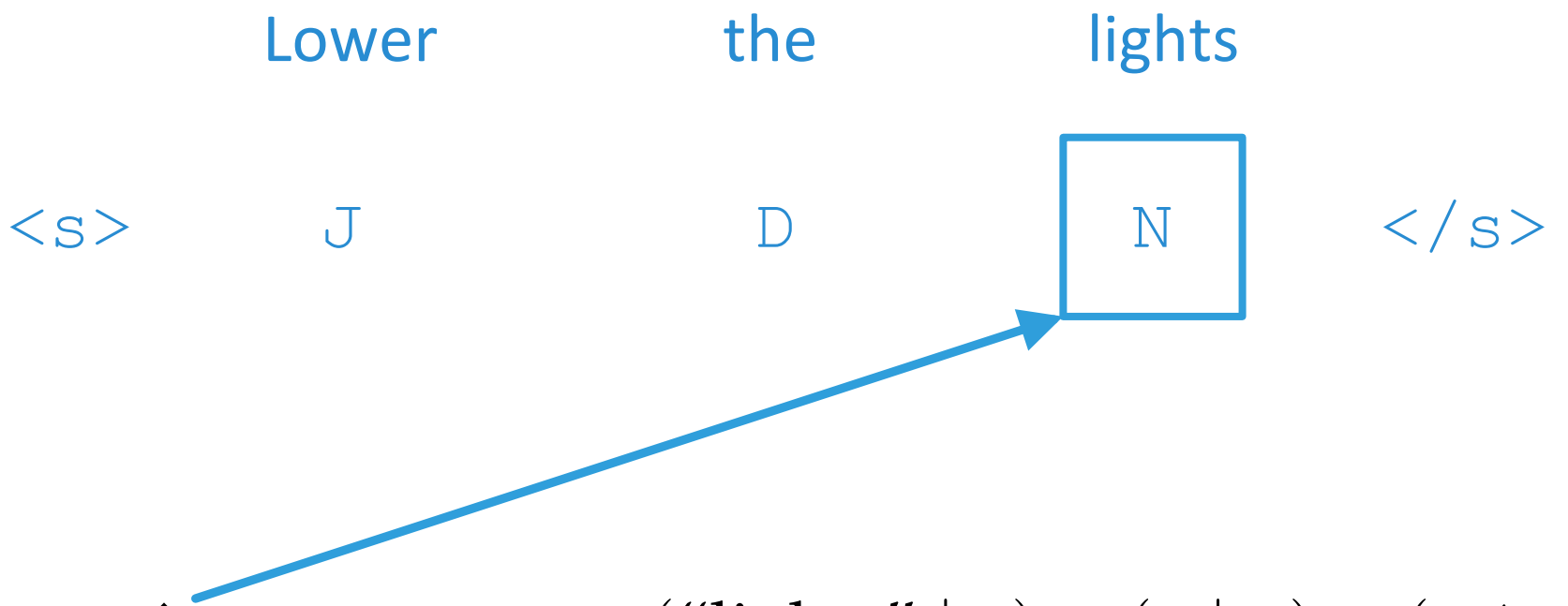
- Assignment 3 has been posted, due two weeks from tomorrow

Inference with Structured Predictors

inference: solve argmax

$$\text{classify}(\boldsymbol{x}, \boldsymbol{\theta}) = \underset{\boldsymbol{y}}{\text{argmax}}\ \text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta})$$

Greedy Inference



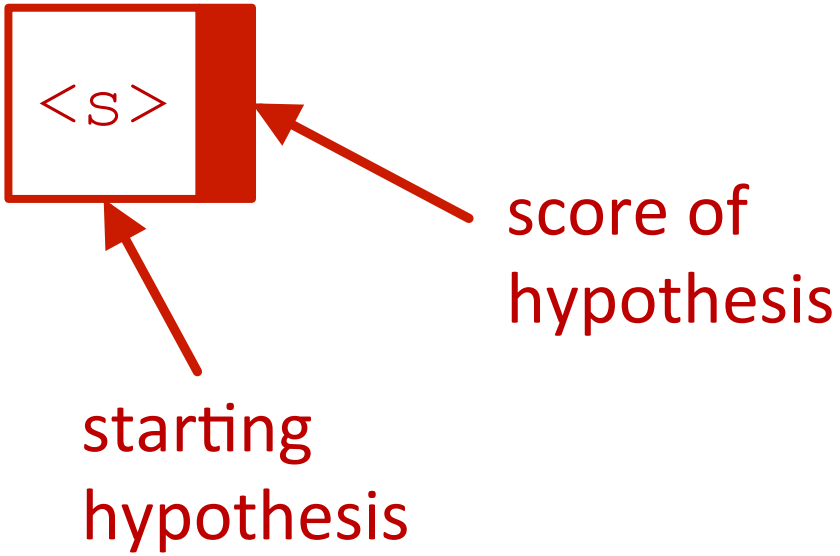
$$\hat{y}_3 = \underset{y \in \mathcal{L}}{\operatorname{argmax}} \; p_{\boldsymbol{\eta}}(\text{“lights”} \mid y)\, p_{\boldsymbol{\tau}}(y \mid \texttt{D})\, p_{\boldsymbol{\tau}}(\texttt{</s>} \mid y)$$

V D N

Lower the lights

Beam Search (beam size $b = 2$)

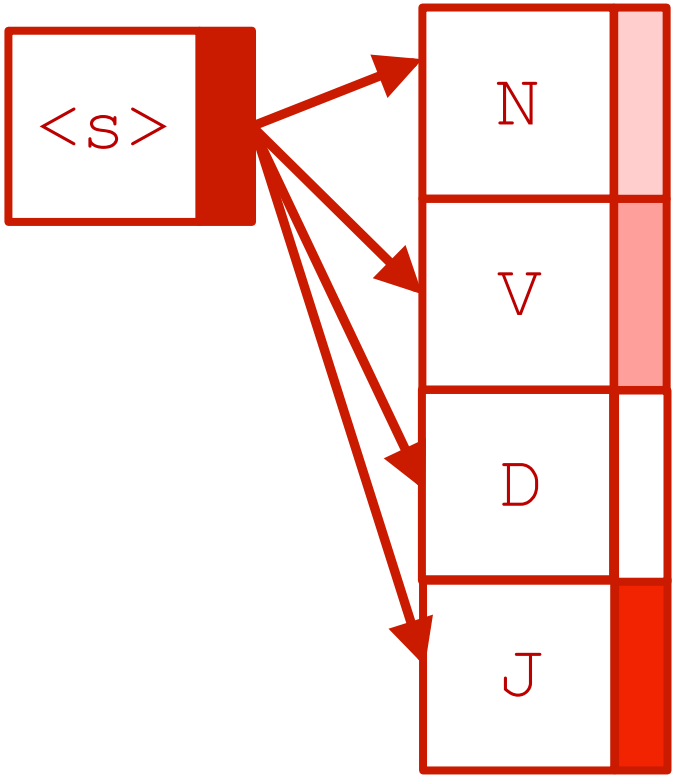
Lower the lights



low score high score

Extend Hypotheses

Lower the lights



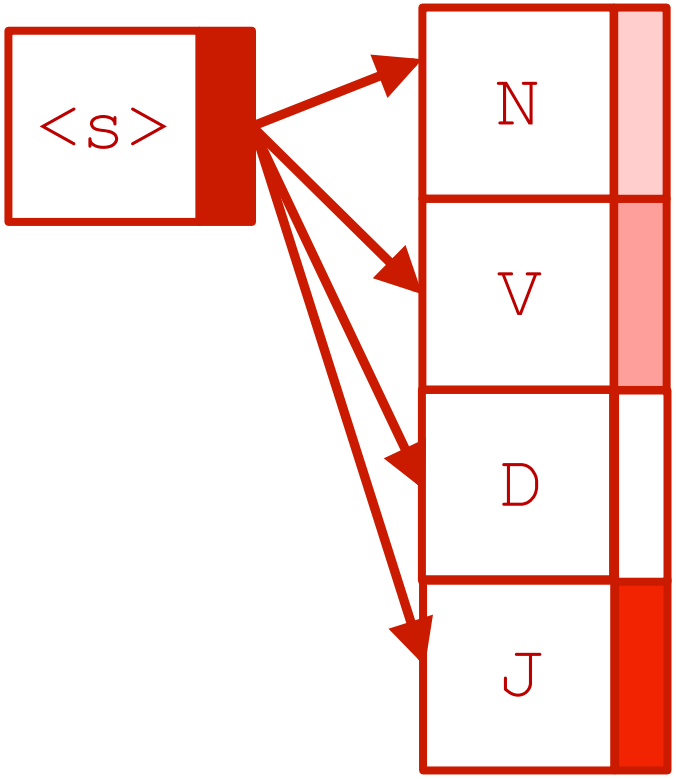
scores of extended hypotheses:

$$p_{\boldsymbol{\eta}}(\text{“Lower”} \mid y)\, p_{\boldsymbol{\tau}}(y \mid \texttt{<s>})\, \text{score}(\text{hyp}_{\text{prev}})$$

low score high score

Prune Hypotheses (b = 2)

Lower the lights

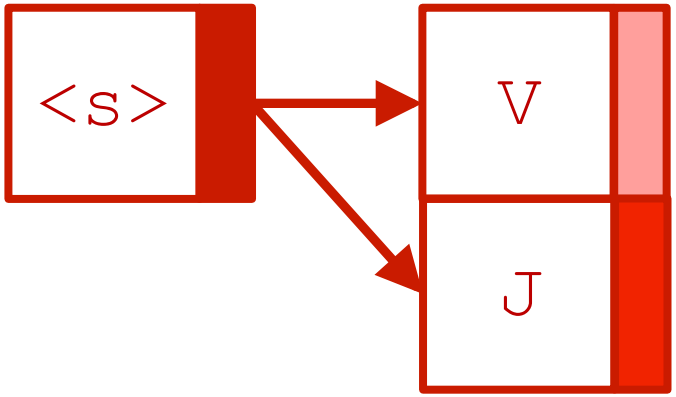


keep top b hypotheses

low score high score

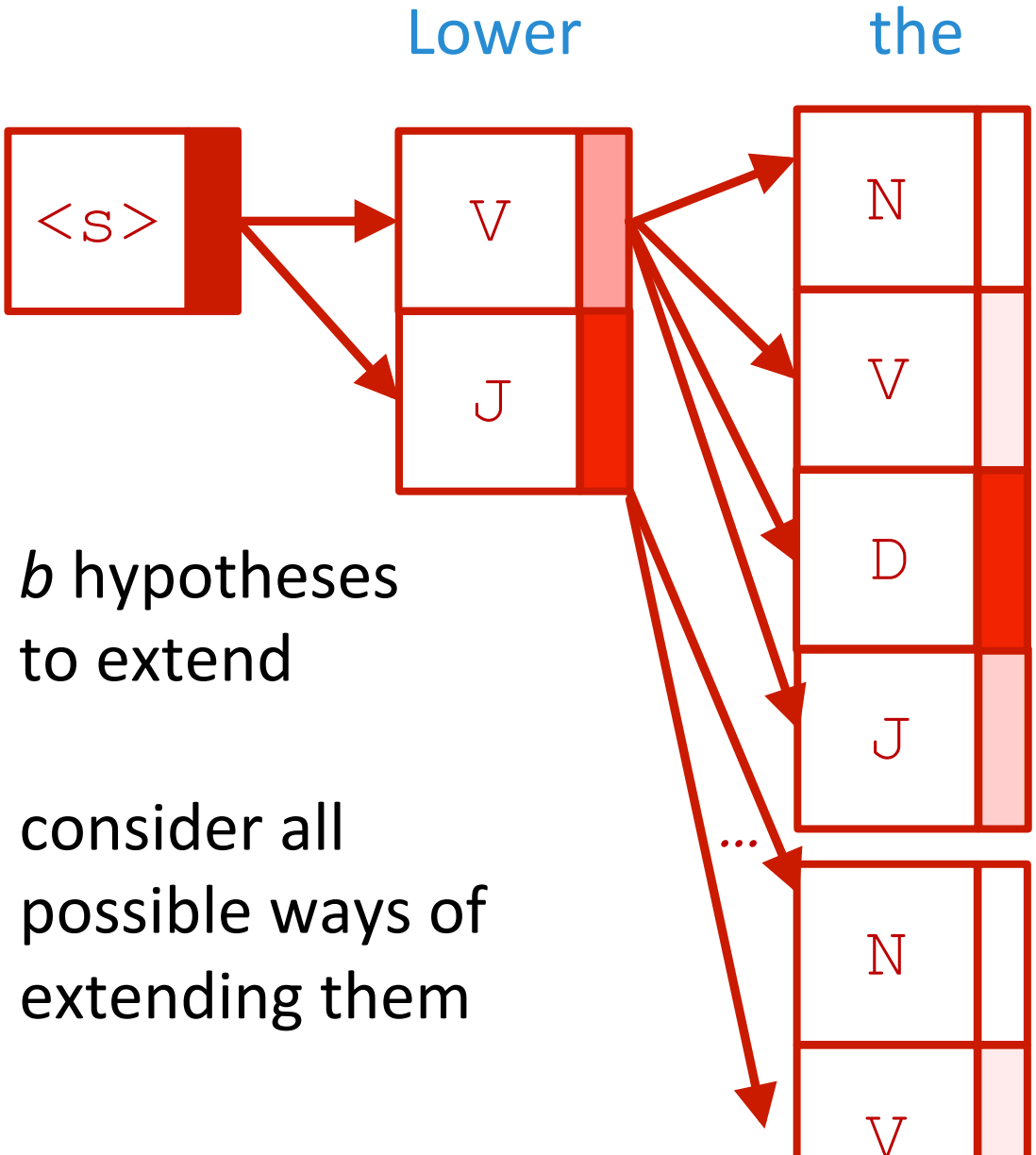
Prune Hypotheses (b = 2)

Lower the lights



low score high score

Extend Hypotheses



b hypotheses to extend

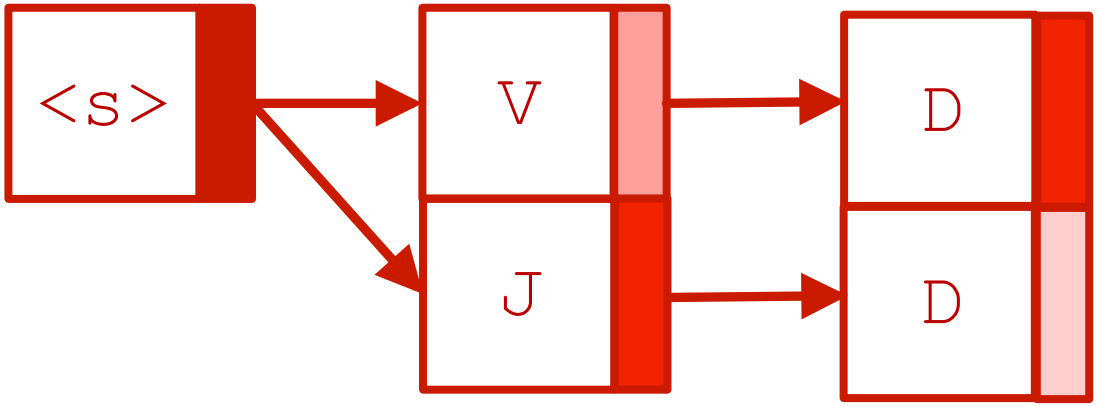
consider all possible ways of extending them

scores of extended hypotheses:

$$p_{\boldsymbol{\eta}}(\text{“the”} \mid y) \times p_{\boldsymbol{\tau}}(y \mid y_{\text{prev}}) \times \text{score}(\text{hyp}_{\text{prev}})$$

Prune Hypotheses (b = 2)

Lower the lights

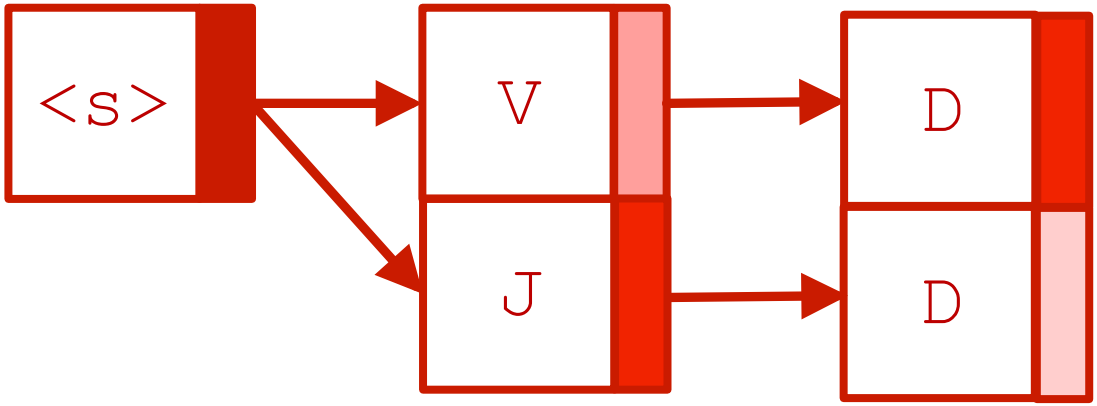


note: due to the small size of HMM parts, these two hypotheses will look identical going forward

we don't need to keep both of them! (unless we're trying to return an n-best list)

Prune Hypotheses (b = 2)

Lower the lights

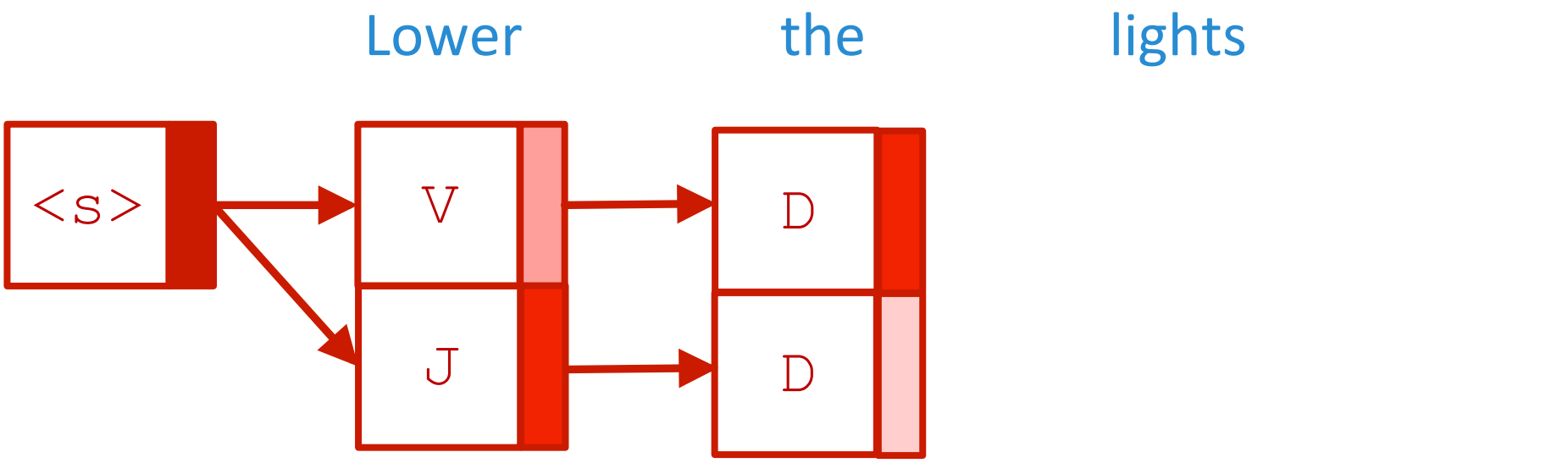


note: due to the small size of HMM parts, these two hypotheses will look identical going forward

we don't need to keep both of them! (unless we're trying to return an n-best list)

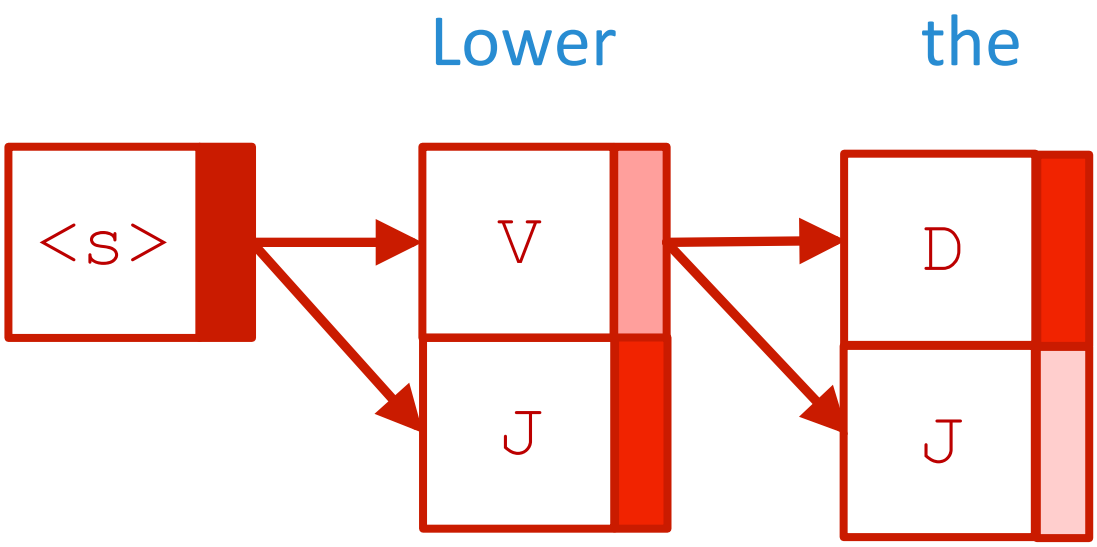
we can use **recombination**

Recombination



recombination: combine hypotheses with identical final labels, keep backpointer for higher-scoring hypothesis

Recombination

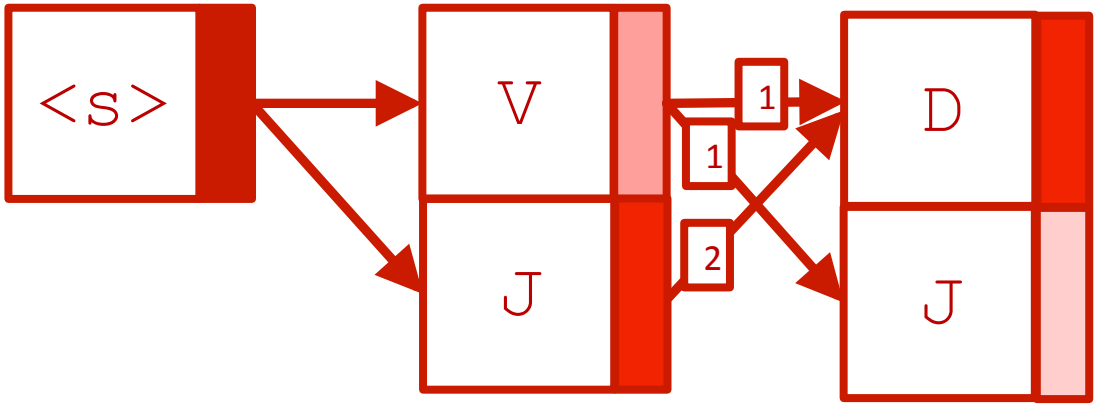


recombination can make better use of hypothesis set

above, we now have space to add another hypothesis and increase label diversity at this position

Recombination with Ranked Backpointers

Lower the lights



we could keep both backpointers, distinguishing them according to score rank

this adds overhead, but could help us later in beam search

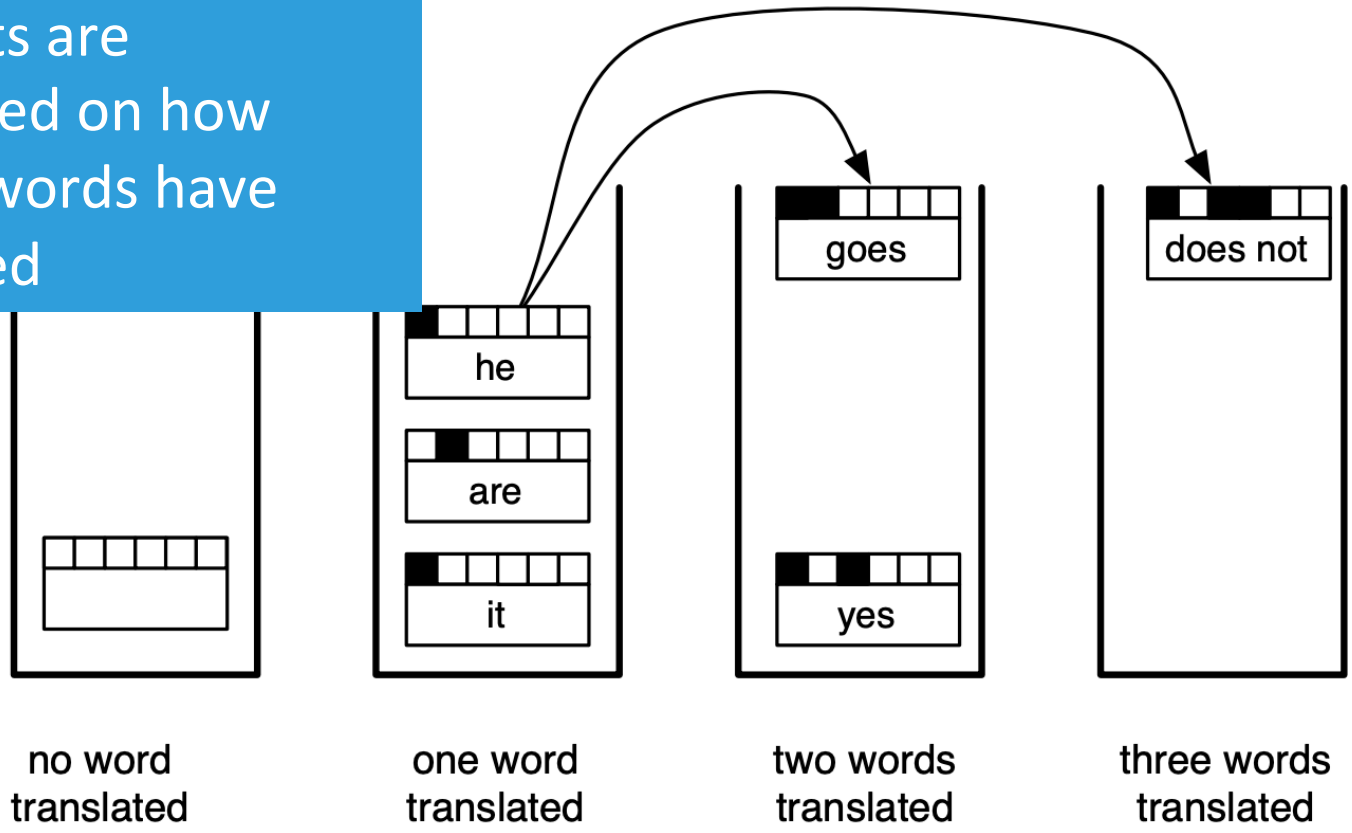
commonly done if we want (approximate) *n*-best lists

Recombination in Beam Search

- recombination has been successfully used in beam search for phrase-based machine translation
- incurs additional overhead, but improves hypothesis diversity
- exact (lossless) recombination only feasible with small parts
- as with other beam search components (extend hypotheses / prune hypothesis set), recombination must be defined by modeler for particular problem / score function in use

Beam Search for Phrase-Based Machine Translation

hypothesis sets are separated based on how many source words have been translated

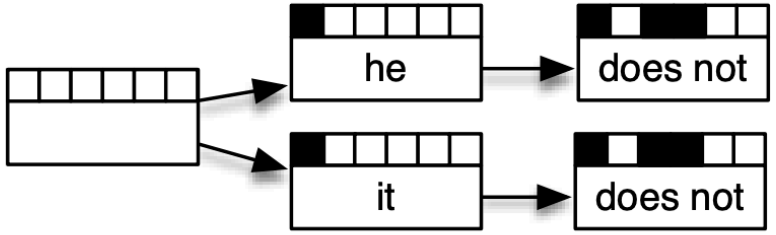


- Hypothesis expansion in a stack decoder
  - translation option is applied to hypothesis
  - new hypothesis is dropped into a stack further down

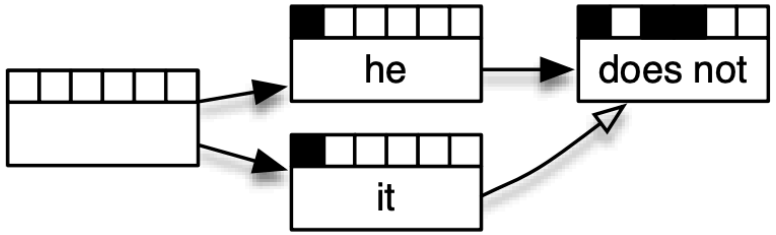
Koehn: *Statistical Machine Translation*

Recombination for Phrase-Based Machine Translation

- Two hypothesis paths lead to hypotheses indistinguishable in subsequent search
  - same number of foreign words translated
  - same last two English words in output (assuming trigram language model)
  - same last foreign word translated
  - different scores



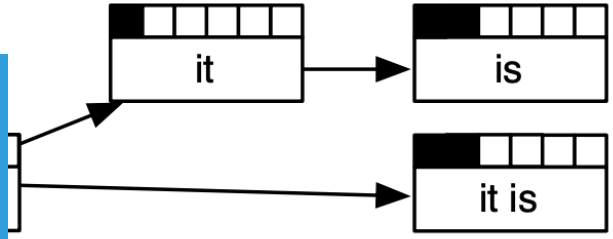
- Worse hypothesis is dropped



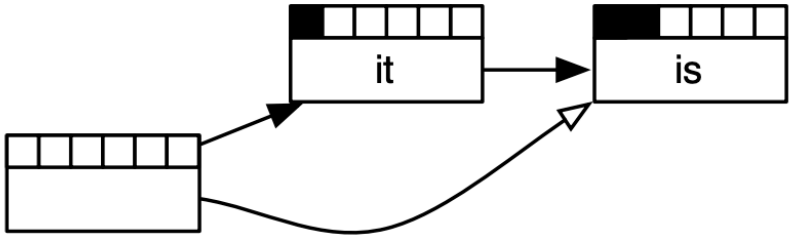
Recombination for Phrase-Based Machine Translation

- Two hypothesis paths lead to two matching hypotheses
  - same number of foreign words translated
  - same English words in the output
  - different scores

this situation ("spurious ambiguity") sometimes arises in structured prediction tasks



- Worse hypothesis is dropped



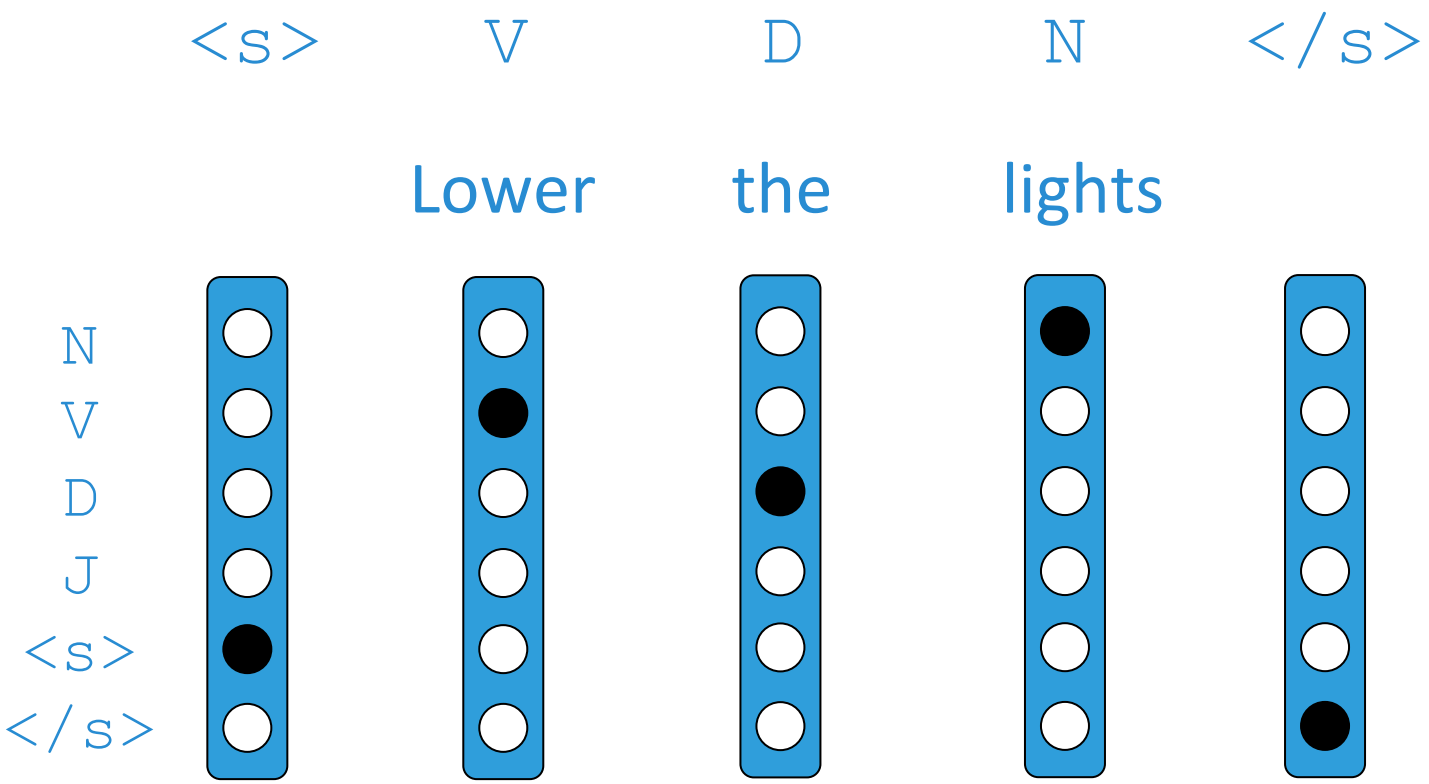
Koehn: *Statistical Machine Translation*

Gradient Descent for Inference

$$\text{classify}(\boldsymbol{x}, \boldsymbol{\theta}) = \underset{\boldsymbol{y}}{\text{argmax}} \ \text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta})$$

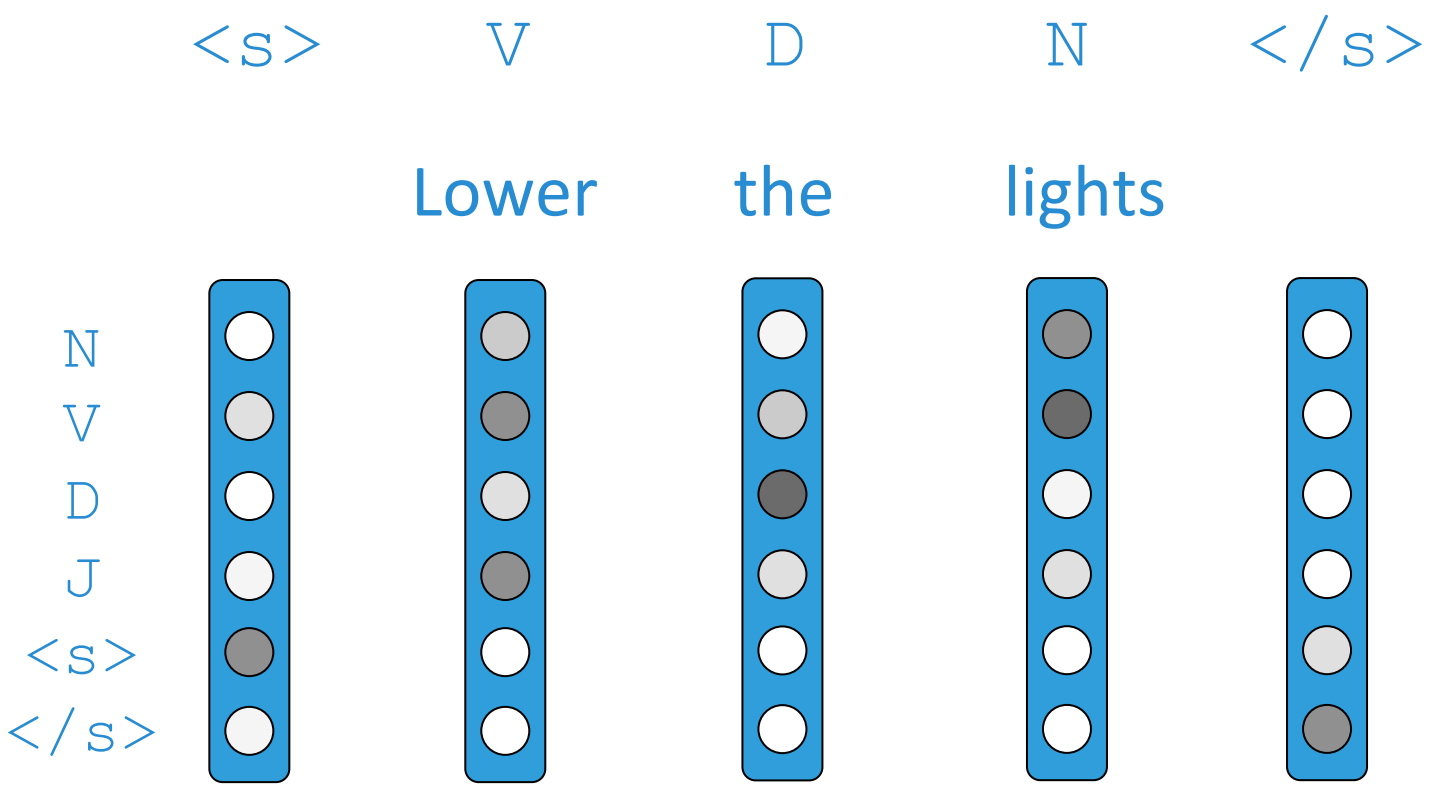
- note: we're talking about using gradient descent for *inference*, not learning
- in solving this argmax, we are doing optimization (over a discrete, structured space)
- we can relax the output space from a discrete space to a continuous one, then apply any optimization method we want (e.g., gradient descent)
- we just need ways to relax the discrete space and then ways to convert the "continuous structured output" to a discrete one after optimization

Relaxing Discrete Output Space for POS Tagging



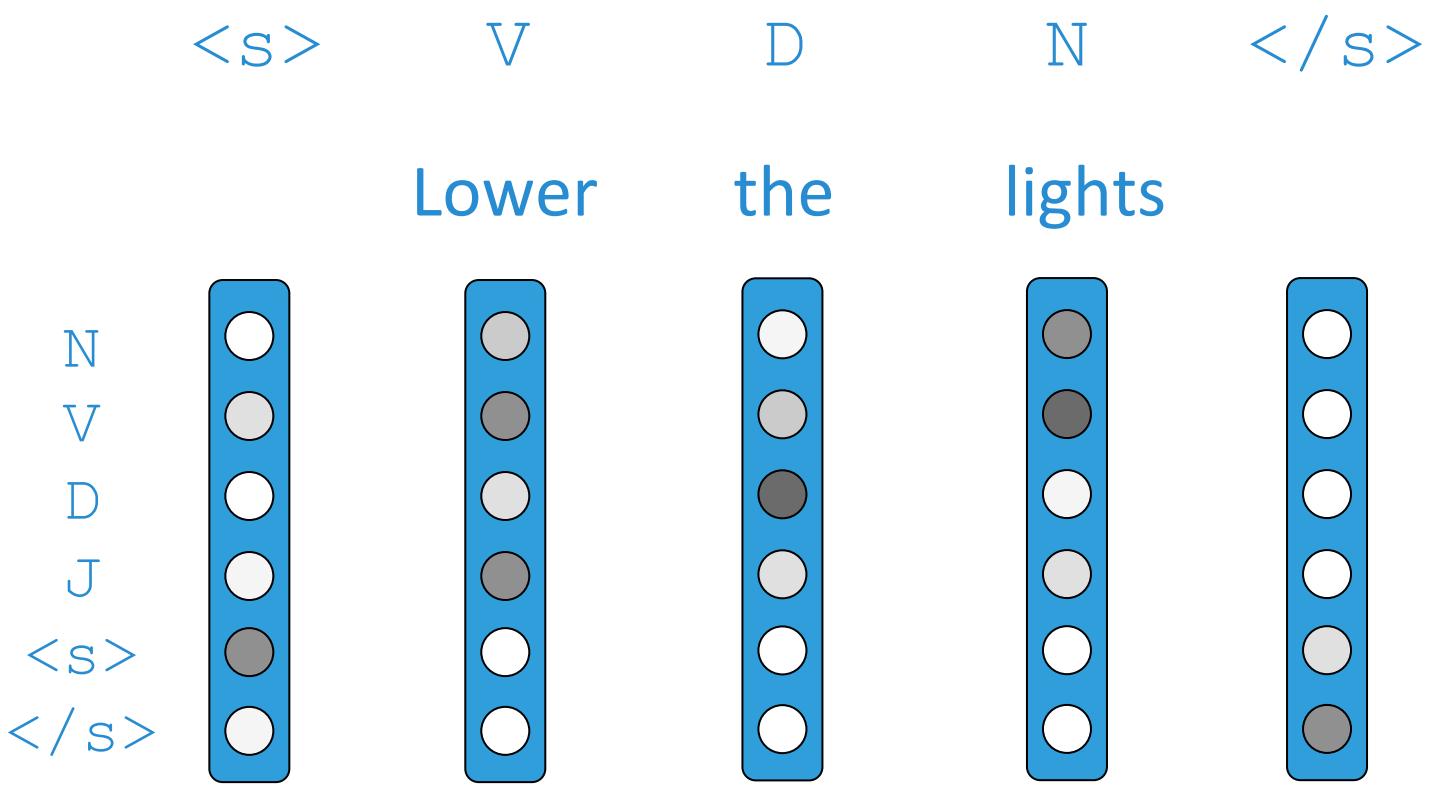
first, represent discrete, structured output using one-hot vectors

Relaxing Discrete Output Space for POS Tagging



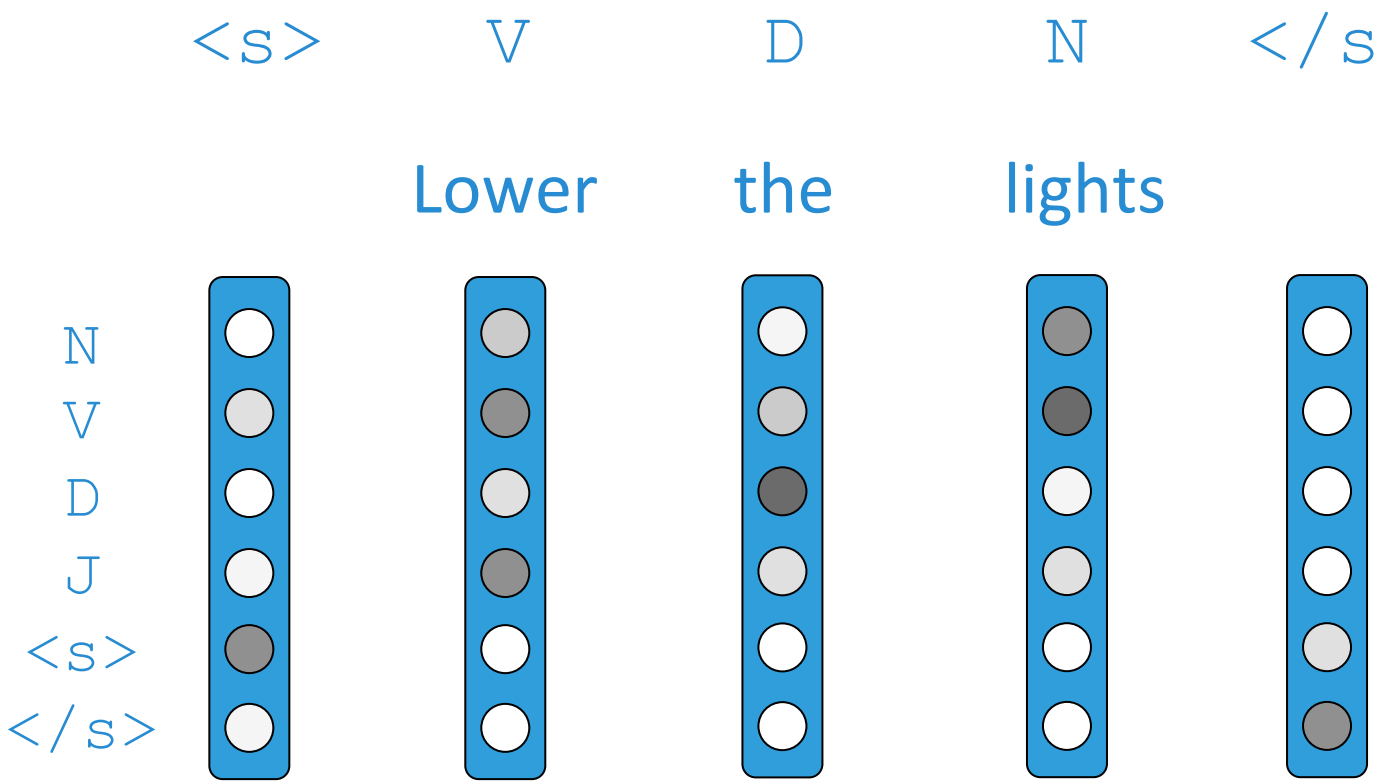
then, relax to vectors where sum of entries is 1 (treat them as distributions over tags)

Relaxing Discrete Output Space for POS Tagging



use gradient descent to optimize the pretrained score function with respect to these vector entries

Relaxing Discrete Output Space for POS Tagging

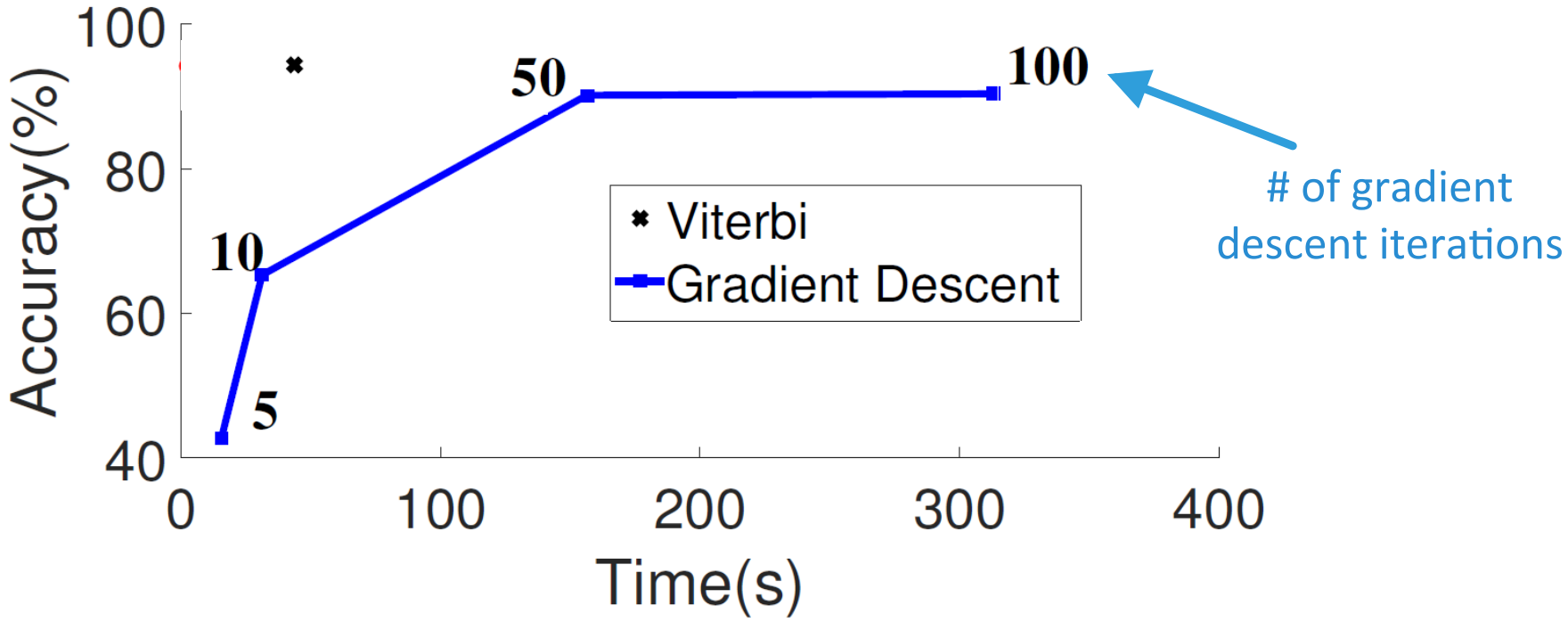


at convergence, find largest value in each vector and return it as the predicted POS tag at that position

Gradient Descent for Inference

- simple and general
- easy to implement thanks to toolkits with automatic differentiation
- it does require a way to relax the output space to be continuous
- how well does it work?

Empirical Comparison



- sequence labeling task with 400 labels (CCG supertagging)
- model is a BLSTM-CRF
- gradient descent is much worse than Viterbi!

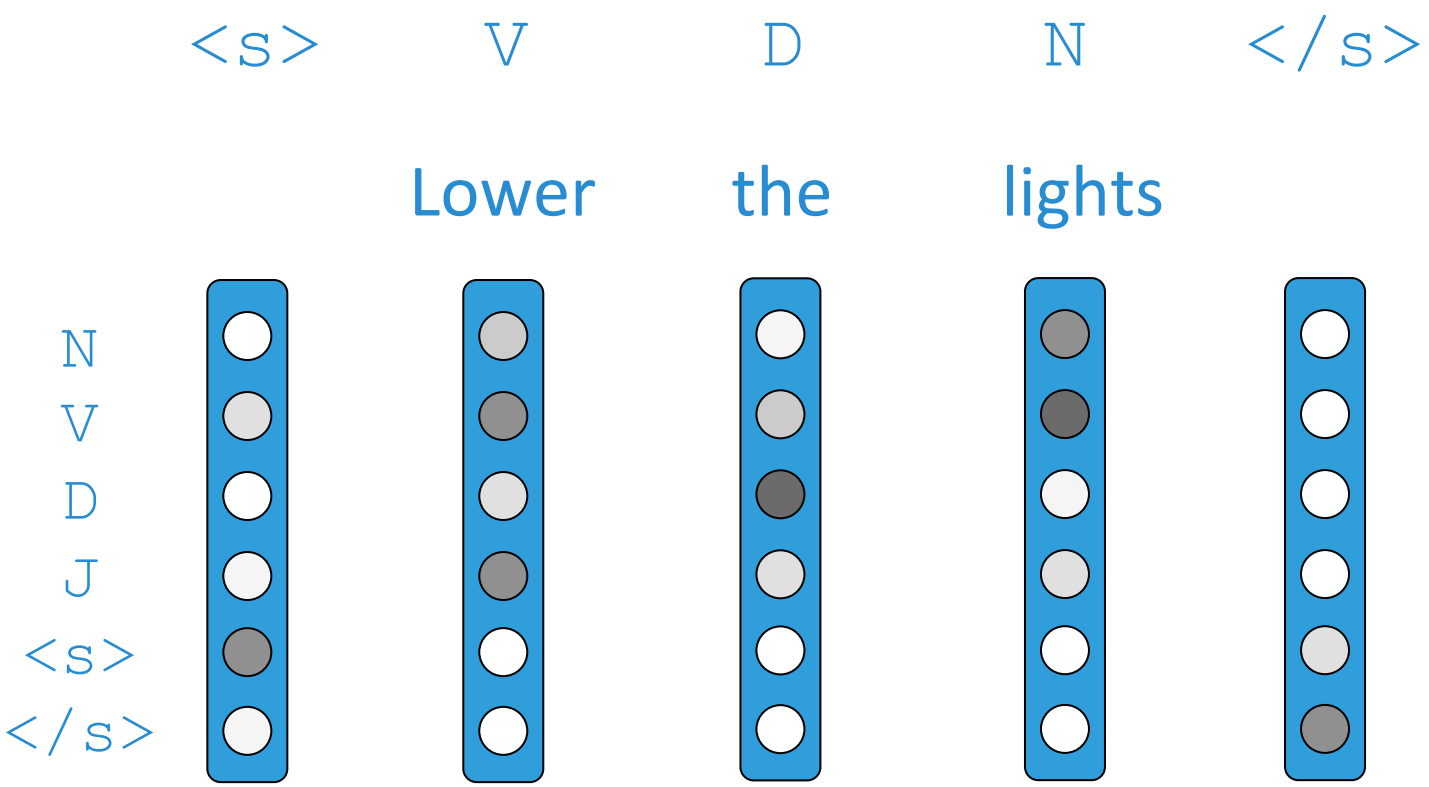
Tu & Gimpel (2019): *Benchmarking Approximate Inference Methods for Neural Structured Prediction*

Inference Networks for Structured Prediction

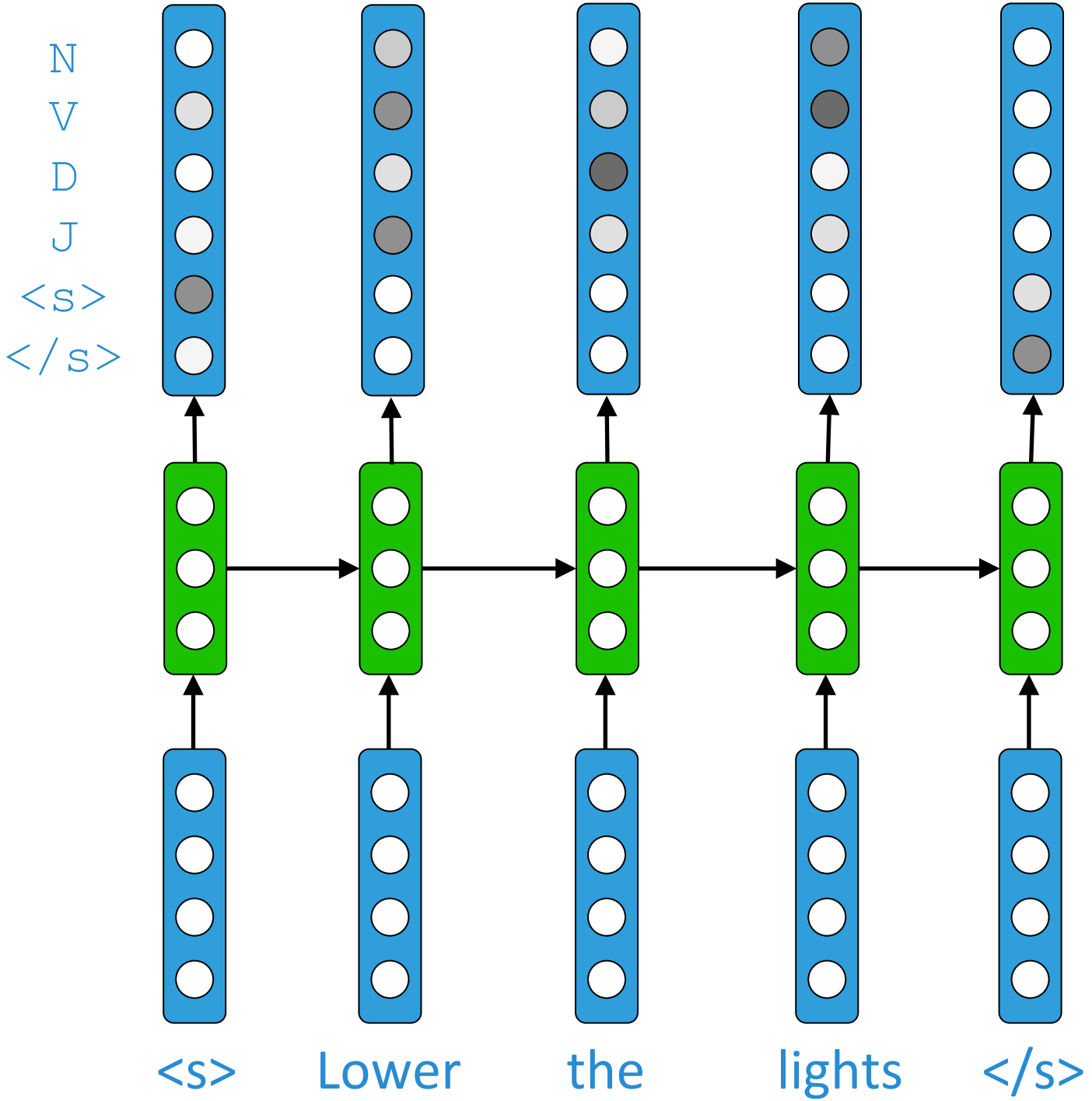
- working in the relaxed continuous output space, we can design a neural network architecture to map from inputs to (relaxed) outputs
- train the network to output a structure with high score (similar to teacher-student networks / knowledge distillation)

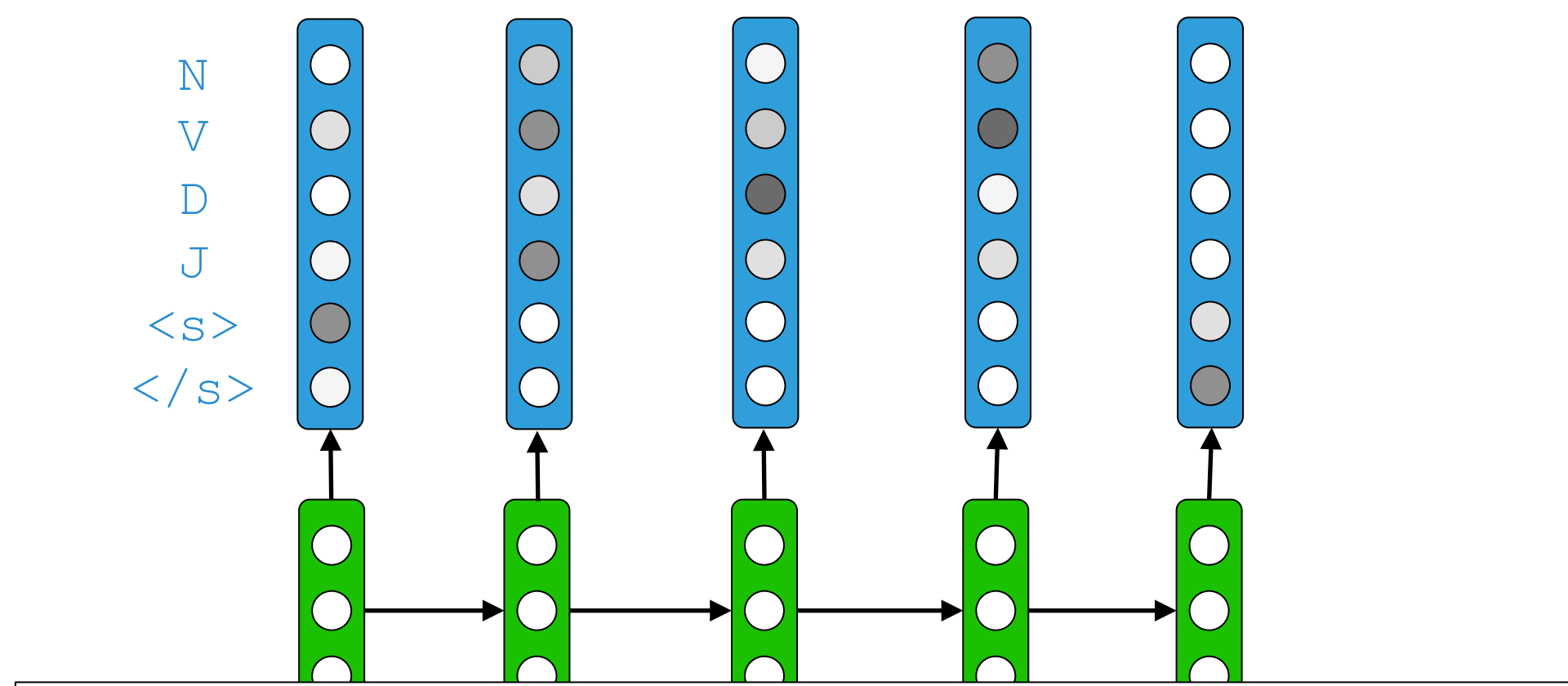
Tu & Gimpel (2018): *Learning Approximate Inference Networks for Structured Prediction*

Inference Networks for POS Tagging



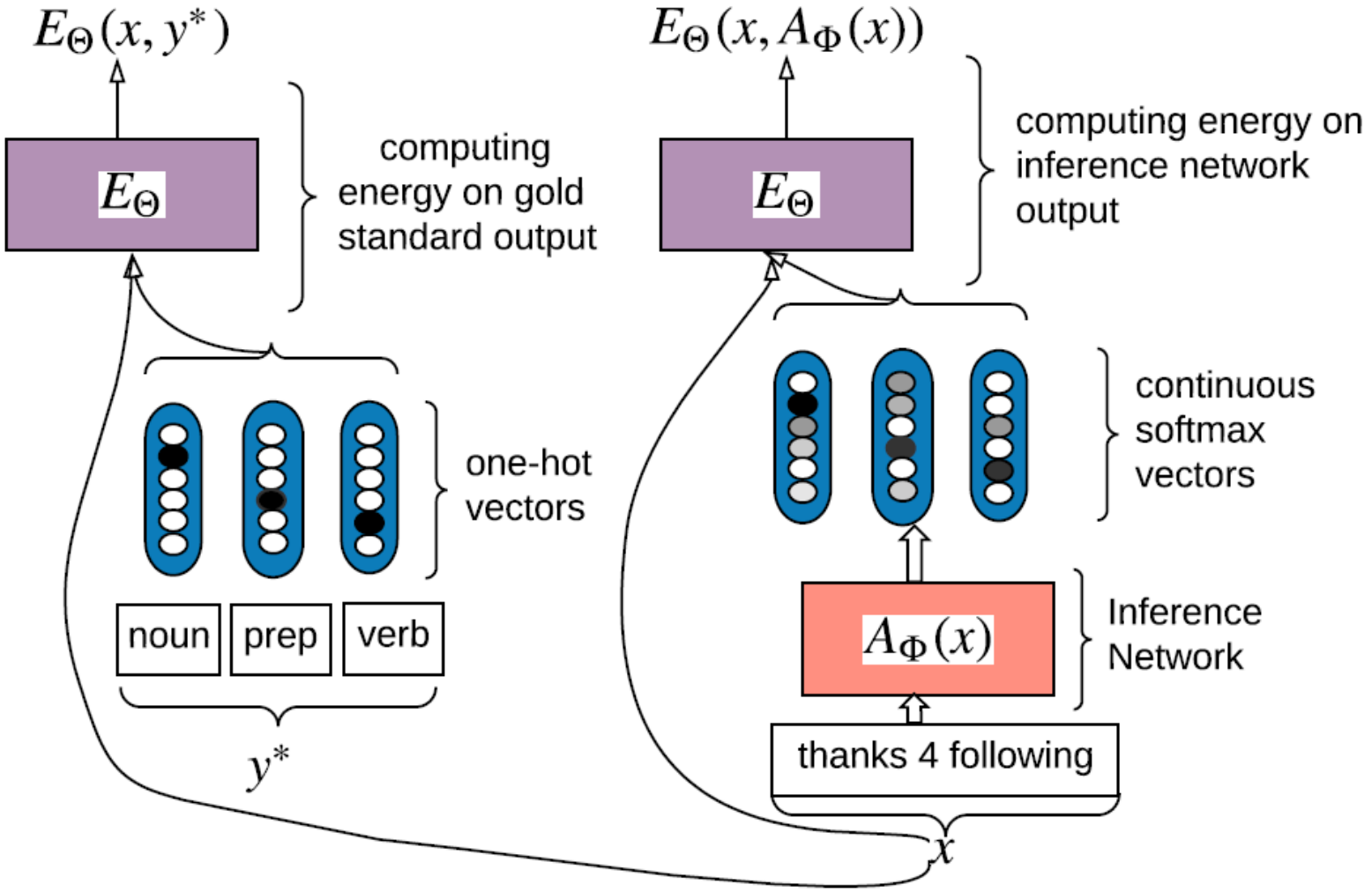
use an LSTM (for example) to go from the sentence to the POS tag distribution at each position


N
V
D
J
<s>
</s>
<s>
Lower
the
lights
</s>



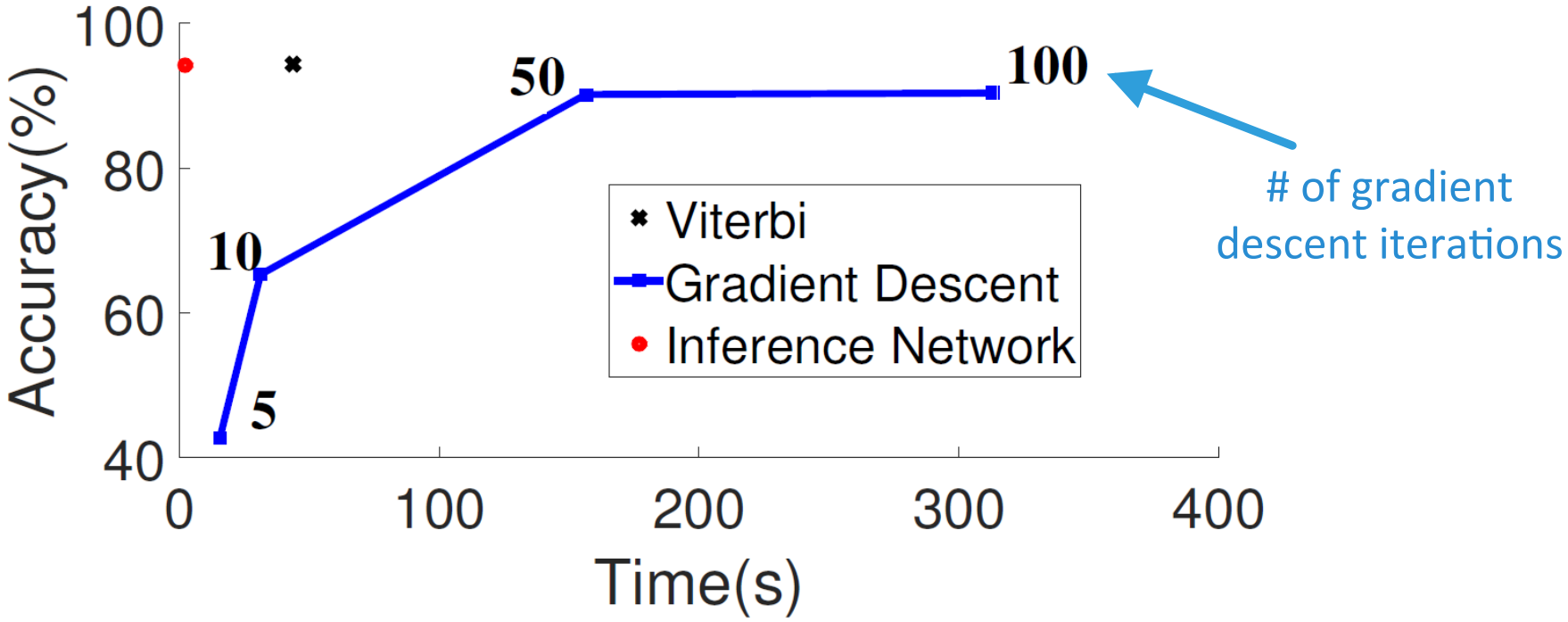
train these LSTM parameters to produce continuous structured outputs with high score under pretrained structured score function

Inference Networks



Tu & Gimpel (2018): *Learning Approximate Inference Networks for Structured Prediction*

Empirical Comparison



- sequence labeling task with 400 labels (CCG supertagging)
- model is a BLSTM-CRF
- inference network shows a much better speed/accuracy trade-off

Tu & Gimpel (2019): *Benchmarking Approximate Inference Methods for Neural Structured Prediction*

Integer Linear Programming

- we can often formulate inference as optimizing an integer linear program
- we can then use off-the-shelf ILP solvers
- sometimes we can relax the ILP to an LP (remove integer constraints), then solve the LP which can be done efficiently, then convert the relaxed structure to a discrete one

Inference: Summary

- exact DP algorithms if parts are small
- beam search
  - can improve with heuristics ("heuristic search")
  - can handle non-local features / large parts
  - recombination can help, though not with large parts
- coarse-to-fine
- gradient descent for inference
- inference networks
- linear programming / ILP

Learning in Structured Prediction

$$\text{classify}(\boldsymbol{x}, \boldsymbol{\theta}) = \underset{\boldsymbol{y}}{\text{argmax}} \ \text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta})$$

learning: choose $\boldsymbol{\theta}$

- most loss functions used in structured prediction have the same form as those used in multi-class classification
- part that changes: now structured inference is required for computing gradients
- we can use any inference strategy we discussed in the context of learning
- there are also new inference problems that arise for certain loss functions

Cost Functions

- **cost function**: how different are these two structures?

$$\text{cost} : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$$

- typically used to compare predicted structure to gold standard
- should reflect evaluation metric for task

- usual conventions: $\text{cost}(\boldsymbol{y}, \boldsymbol{y}) = 0$

$$\text{cost}(\boldsymbol{y}, \boldsymbol{y}') = \text{cost}(\boldsymbol{y}', \boldsymbol{y})$$

Cost Functions

- typical cost for multi-class classification:

$$\text{cost}(y, y') = \mathbb{I}[y \neq y']$$

- how about for sequences? $\text{cost} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$
  - "Hamming cost": $\text{cost}(\boldsymbol{y}, \boldsymbol{y}') = \sum_{t=1}^{|\boldsymbol{y}|} \mathbb{I}[y_t \neq y'_t]$
  - "0-1 cost": $\text{cost}(\boldsymbol{y}, \boldsymbol{y}') = \mathbb{I}[\boldsymbol{y} \neq \boldsymbol{y}']$

Empirical Risk Minimization

$$\hat{\boldsymbol{\theta}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \sum_{\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{D}} \text{cost}(\boldsymbol{y}, \text{predict}(\boldsymbol{x}, \boldsymbol{\theta}))$$

$$\text{predict}(\boldsymbol{x}, \boldsymbol{\theta}) = \operatorname*{argmax}_{\boldsymbol{y}} \text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta})$$

- this is intractable so we typically minimize a **surrogate loss function** instead

Loss Functions for Structured Prediction

name	loss	where used
cost ("0-1")	$\text{cost}(\boldsymbol{y}, \text{predict}(\boldsymbol{x}, \boldsymbol{\theta}))$	MERT (Och, 2003)

Loss Functions for Structured Prediction

name	loss	where used
cost ("0-1")	$\text{cost}(\boldsymbol{y}, \text{predict}(\boldsymbol{x}, \boldsymbol{\theta}))$	MERT (Och, 2003)
percep-tron	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} \text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})$	structured perceptron (Collins, 2002)

Loss Functions for Structured Prediction

name	loss	where used
cost ("0-1")	$\text{cost}(\boldsymbol{y}, \text{predict}(\boldsymbol{x}, \boldsymbol{\theta}))$	MERT (Och, 2003)
percep-tron	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} \text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})$	structured perceptron (Collins, 2002)
hinge	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} (\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}, \boldsymbol{y}'))$	structured SVMs (Taskar et al., *inter alia*)

Max-Margin Markov Networks

Max-Margin Markov Networks

Ben Taskar **Carlos Guestrin** **Daphne Koller**
{*btaskar,guestrin,koller*}*@cs.stanford.edu*
Stanford University

Abstract

In typical classification tasks, we seek a function which assigns a label to a single object. Kernel-based approaches, such as support vector machines (SVMs), which maximize the margin of confidence of the classifier, are the method of choice for many such tasks. Their popularity stems both from the ability to use high-dimensional feature spaces, and from their strong theoretical guarantees. However, many real-world tasks involve sequential, spatial, or structured data, where multiple labels must be assigned. Existing kernel-based methods ignore structure in the problem, assigning labels independently to each object, losing much useful information. Conversely, probabilistic graphical models, such as Markov networks, can represent correlations between labels, by exploiting problem structure, but cannot handle high-dimensional feature spaces, and lack strong theoretical generalization guarantees. In this paper, we present a new framework that combines the advantages of both approaches: *Maximum margin Markov (M^3) networks* incorporate both kernels, which efficiently deal with high-dimensional features, and the ability to capture correlations in structured data. We present an efficient algorithm for learning M^3 networks based on a

Maximum-Margin Markov Networks

- parts function contains parts on consecutive labels ("bigram parts")
- arbitrary features of input are permitted
- test-time inference uses Viterbi Algorithm
- learning done by minimizing **hinge** loss (inference algorithms used to compute **subgradients**)

A New Structured Inference Problem

- test-time inference:

$$\underset{\boldsymbol{y}}{\operatorname{argmax}} \ \operatorname{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta})$$

- "cost-augmented" inference includes cost function:

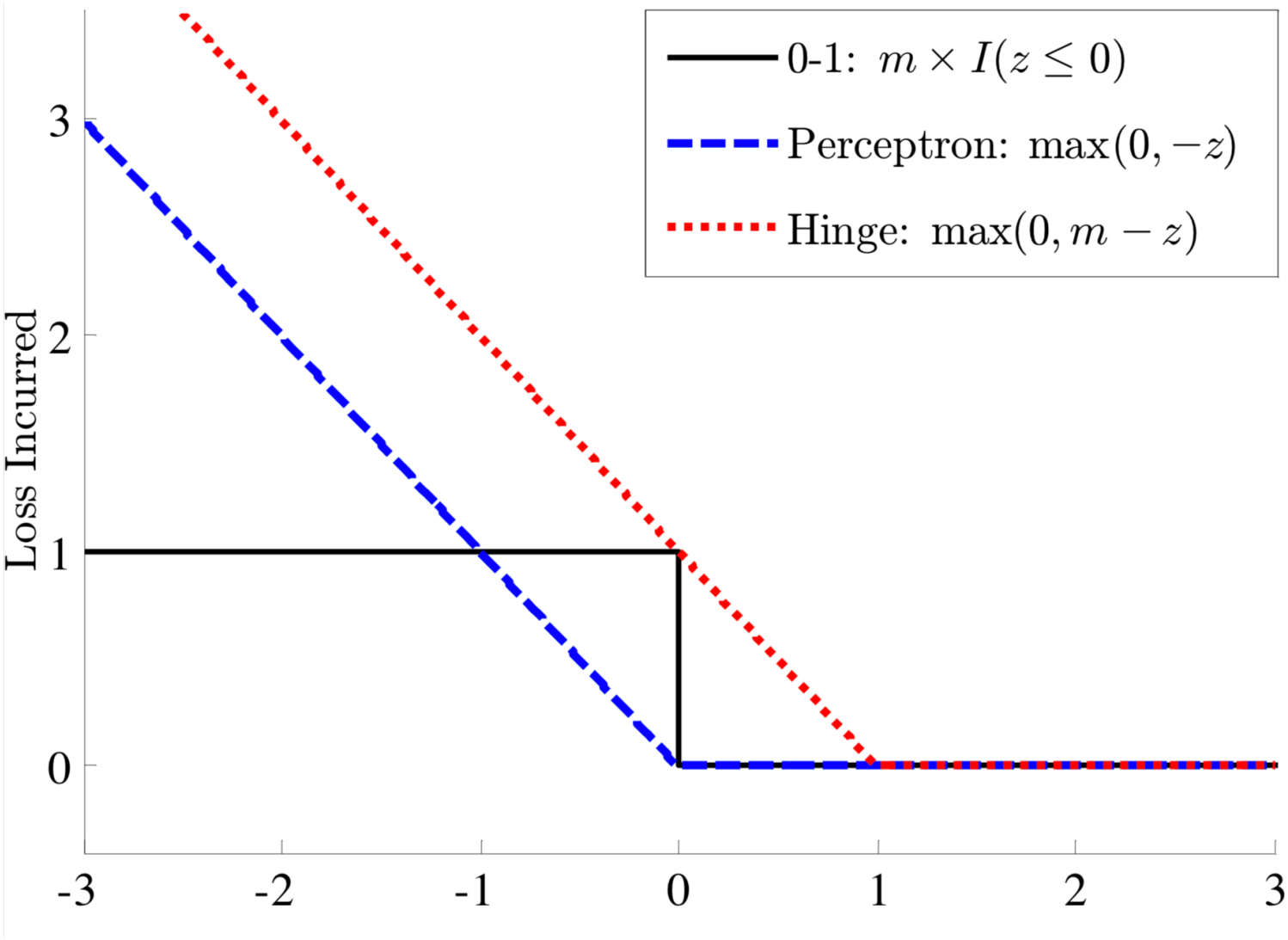
$$\max_{\boldsymbol{y}'} \ (\operatorname{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \operatorname{cost}(\boldsymbol{y}, \boldsymbol{y}'))$$

  - only used during training
  - we can use all the same argmax inference algorithms as for the test-time inference problem as long as the cost function parts are not too big

Loss Functions for Structured Prediction

name	loss	where used
cost ("0-1")	$\text{cost}(\boldsymbol{y}, \text{predict}(\boldsymbol{x}, \boldsymbol{\theta}))$	MERT (Och, 2003)
percep-tron	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} \text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})$	structured perceptron (Collins, 2002)
hinge	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} (\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}, \boldsymbol{y}'))$	structured SVMs (Taskar et al., *inter alia*)

Visualizing Losses for Binary Classification



this is for binary classification, so *y* is either -1 or 1

m = cost multiplier

z = classifier score (the larger it is, the more confident the classifier is)

Loss Functions for Structured Prediction

name	loss	where used
cost ("0-1")	$\text{cost}(\boldsymbol{y}, \text{predict}(\boldsymbol{x}, \boldsymbol{\theta}))$	MERT (Och, 2003)
percep-tron	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} \text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})$	structured perceptron (Collins, 2002)
hinge	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} (\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}, \boldsymbol{y}'))$	structured SVMs (Taskar et al., *inter alia*)
log	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y}'} \exp\{\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})\}$	CRFs (Lafferty et al., 2001)

(Chain) Conditional Random Fields

Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data

John Lafferty[†*] LAFFERTY@CS.CMU.EDU
Andrew McCallum[*†] MCCALLUM@WHIZBANG.COM
Fernando Pereira[*‡] FPEREIRA@WHIZBANG.COM

[*]WhizBang! Labs–Research, 4616 Henry Street, Pittsburgh, PA 15213 USA
[†]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA
[‡]Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104 USA

Abstract

We present *conditional random fields*, a framework for building probabilistic models to segment and label sequence data. Conditional random fields offer several advantages over hidden Markov models and stochastic grammars for such tasks, including the ability to relax strong independence assumptions made in those models. Conditional random fields also avoid a fundamental limitation of maximum entropy Markov models (MEMMs) and other discriminize the joint likelihood of training examples. To define a joint probability over observation and label sequences, a generative model needs to enumerate all possible observation sequences, typically requiring a representation in which observations are task-appropriate atomic entities, such as words or nucleotides. In particular, it is not practical to represent multiple interacting features or long-range dependencies of the observations, since the inference problem for such models is intractable.

This difficulty is one of the main motivations for looking at conditional models as an alternative. A conditional model

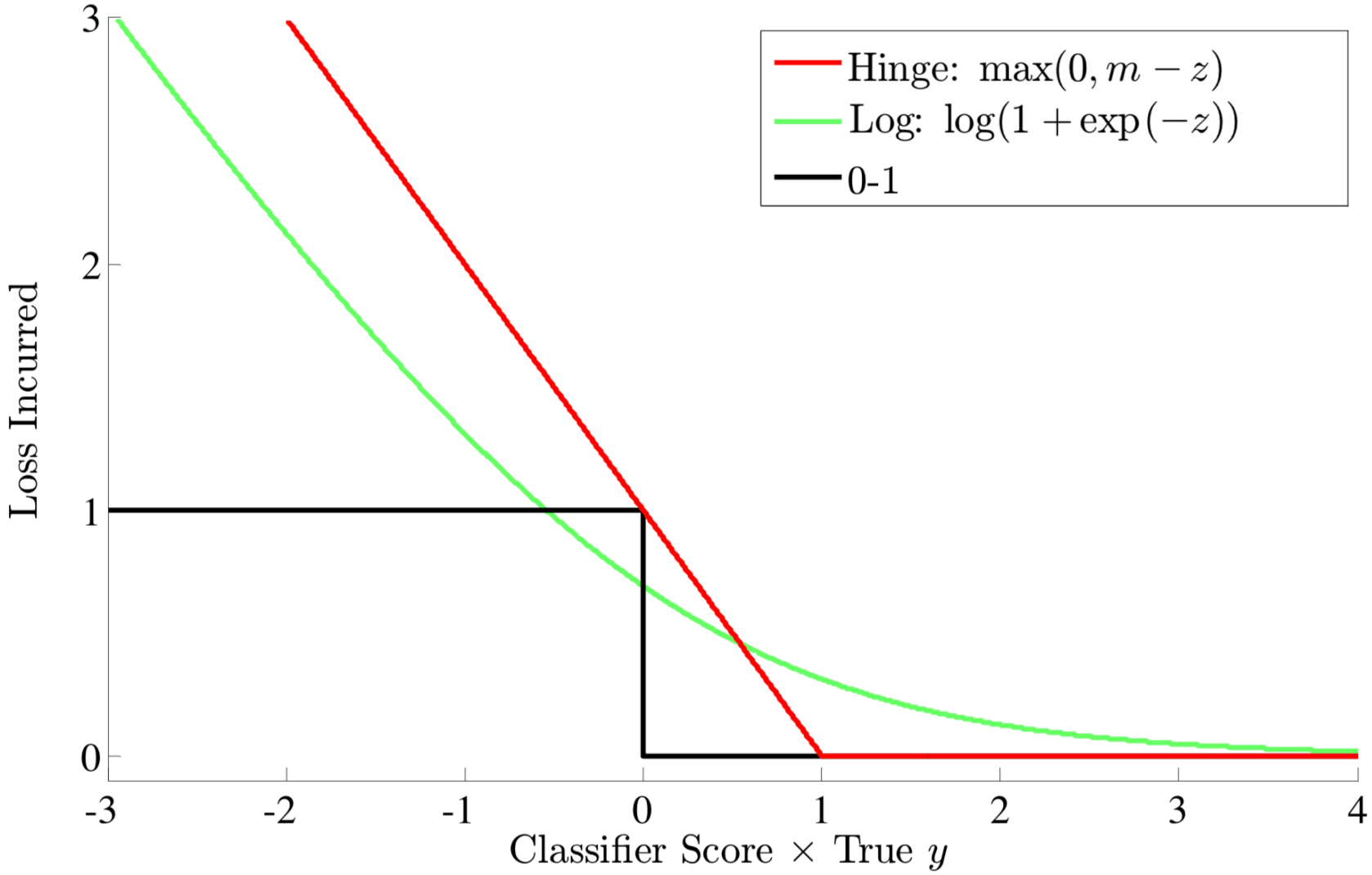
(Chain) Conditional Random Fields

- parts function contains parts on consecutive labels ("bigram parts")
- arbitrary features of input are permitted
- test-time inference uses Viterbi Algorithm
- learning done by minimizing log loss (DP algorithms used to compute gradients)

Loss Functions for Structured Prediction

name	loss	where used
cost ("0-1")	$\text{cost}(\boldsymbol{y}, \text{predict}(\boldsymbol{x}, \boldsymbol{\theta}))$	MERT (Och, 2003)
percep-tron	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} \text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})$	structured perceptron (Collins, 2002)
hinge	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} (\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}, \boldsymbol{y}'))$	structured SVMs (Taskar et al., *inter alia*)
log	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y}'} \exp\{\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})\}$	CRFs (Lafferty et al., 2001)

Visualizing Losses for Binary Classification



this is for binary classification, so *y* is either -1 or 1

m = cost multiplier

z = classifier score (the larger it is, the more confident the classifier is)

Loss Functions for Structured Prediction

name	loss	where used
cost ("0-1")	$\text{cost}(\boldsymbol{y}, \text{predict}(\boldsymbol{x}, \boldsymbol{\theta}))$	MERT (Och, 2003)
percep-tron	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} \text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})$	structured perceptron (Collins, 2002)
hinge	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} (\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}, \boldsymbol{y}'))$	structured SVMs (Taskar et al., *inter alia*)
log	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y}'} \exp\{\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})\}$	CRFs (Lafferty et al., 2001)
softmax margin	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y}'} \exp\{\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}, \boldsymbol{y}')\}$	Povey et al. (2008), Gimpel & Smith (2010)

Relationships Among Losses

$$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} \text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})$$

perceptron loss

max to softmax

$$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y}'} \exp\{\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})\}$$

log loss

add cost function

add cost function

max-margin

max to softmax

softmax-margin

$$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} (\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}, \boldsymbol{y}'))$$

$$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y}'} \exp\{\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}, \boldsymbol{y}')\}$$