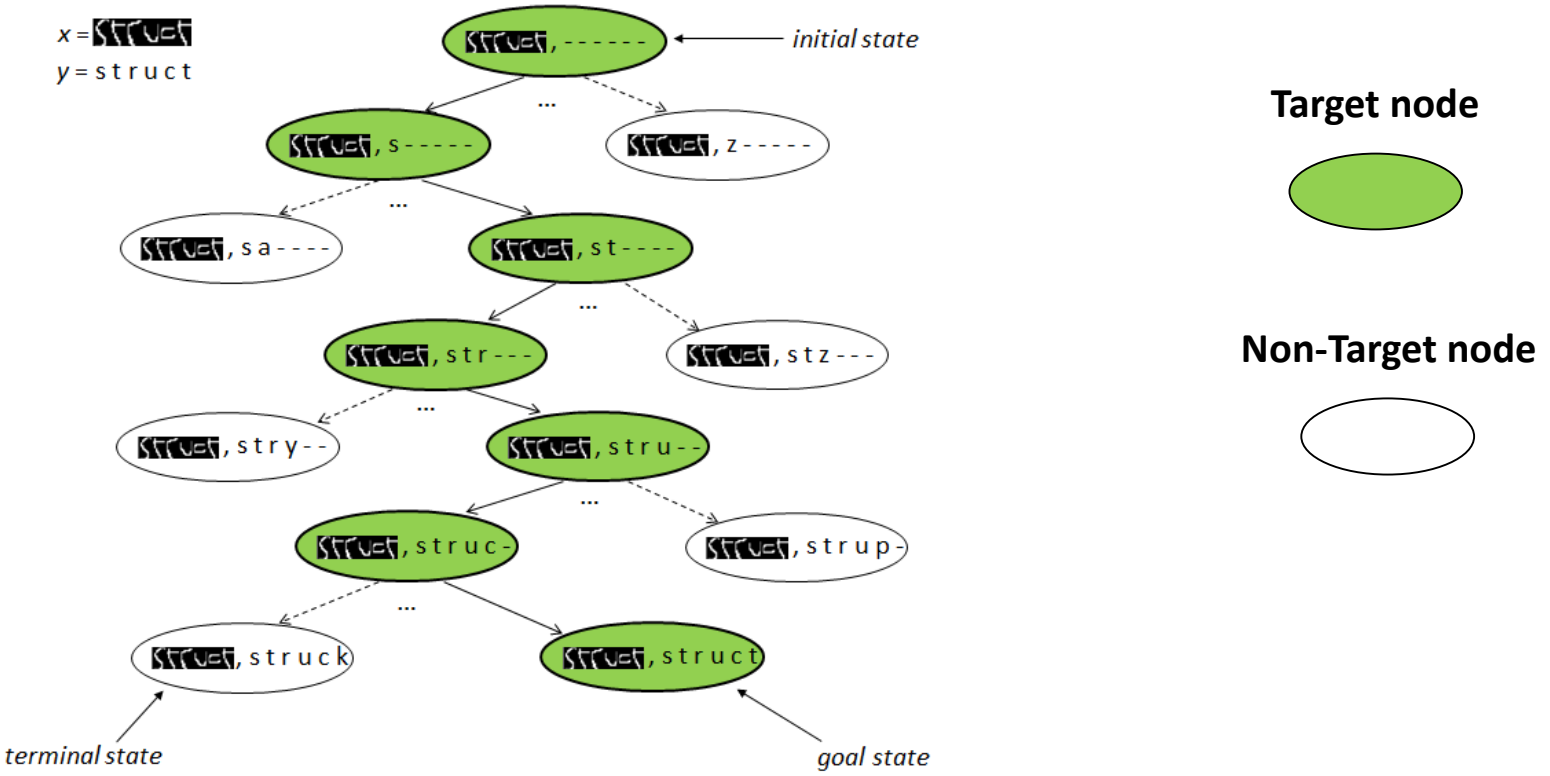
Lecture #4: LaSO Framework

Beam Search Framework

- **Given**
  - Search space definition (ordered or unordered)
  - Training examples (input-output pairs)
  - Beam width B (>1)
- **Learning Goal**
  - Learn a heuristic function to quickly guide the search to the correct "complete" output
- **Key Idea:**
  - Structured prediction as a search problem in the space of partial outputs
  - Training examples define target paths from initial state to the goal state (correct structured output)

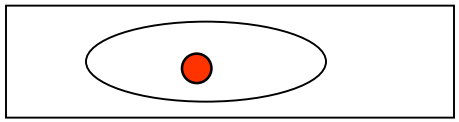
Beam Search Framework: Key Elements

- 1) Search space; 2) Search procedure; 3) **Heuristic function**

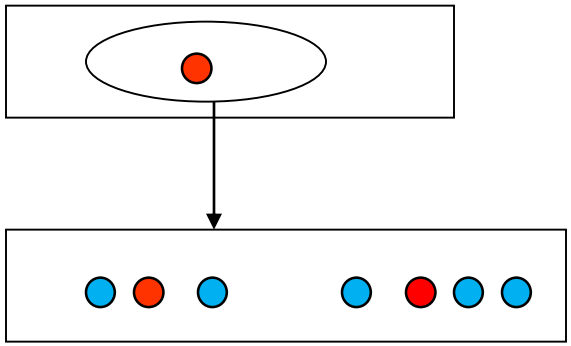


- Represent heuristic function as a linear function
  - $H(n) = w \cdot \psi(n)$, where $\psi(n)$ stands for features of node n

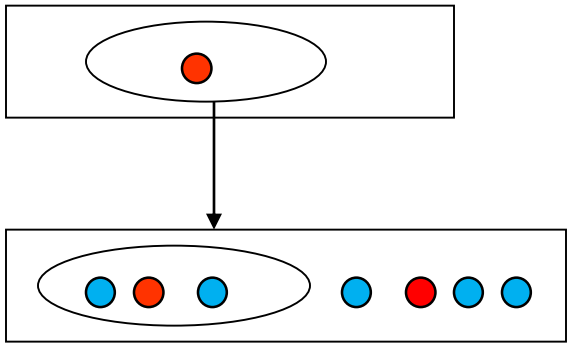
Beam Search: Illustration



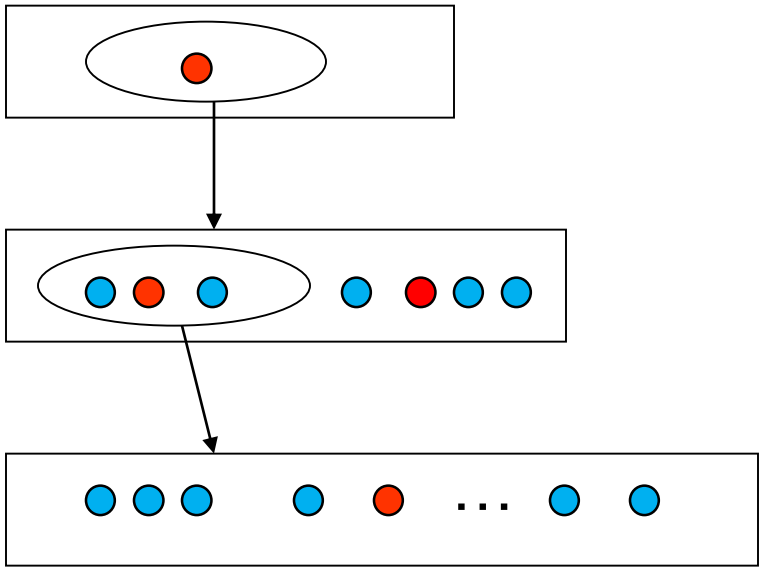
Beam Search: Illustration



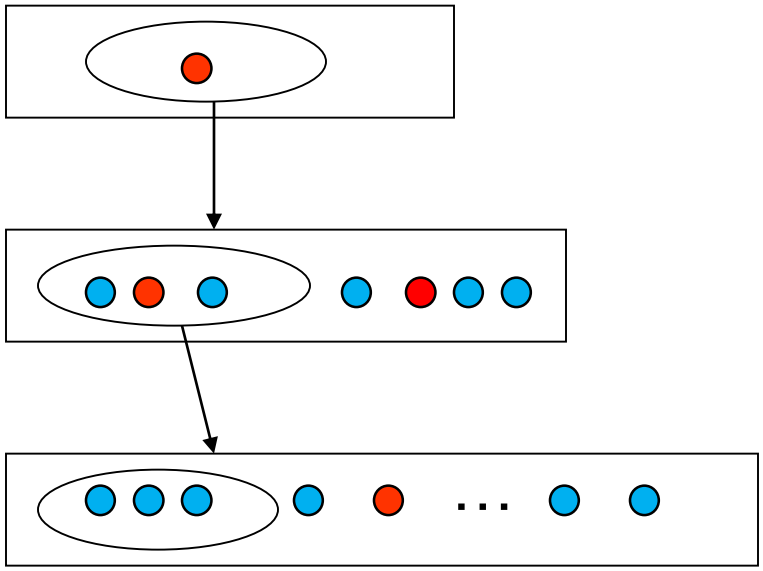
Beam Search: Illustration



Beam Search: Illustration



Beam Search: Illustration



Beam Search Framework: Inference

- **Input:** learned weights w; beam width B; structured input x

- **repeat**
  - Perform search with heuristic $H(n) = w \cdot \psi(n)$
- **until** *reaching a terminal state*

- **Output:** the complete output y corresponding to the terminal state

Beam Search Framework: Generic Learning Template

- **Three design choices**
  - How to define the notion of "search error"?
  - How to "update the weights" of heuristic function when a search error is encountered?
  - How to "update the beam" after weight update?

Beam Search Framework: Learning Instantiations

- Early update

  [Collins and Roark, 2004]

- Max-violation update

  [Huang et al., 2012]

- Learning as Search Optimization (LaSO)

  [Daume et al., 2005], [Xu et al., 2009]

Beam Search Framework: Learning Instantiations

- Early update
- Max-violation update
- Learning as Search Optimization (LaSO)

Beam Search Framework: Early Update

- **Search error:** NO target node in the beam
  - We cannot reach the goal node (correct structured output)
- **Weight update:** standard structured perceptron
  - Score of correct output > score of bad output
- **Beam update:** reset beam with initial state OR discontinue search

Beam Search Framework: Early Update

- **repeat**
  - For every training example (x, y)
    - Perform search with current heuristic (weights)
    - If search error, update weights
    - Reset beam with initial state
    - (Dis)continue search
- **until** *convergence* or *max. iterations*

Beam Search Framework: Learning Instantiations

- Early update
- Max-violation update
- Learning as Search Optimization (LaSO)

Beam Search Framework: Max-Violation Update

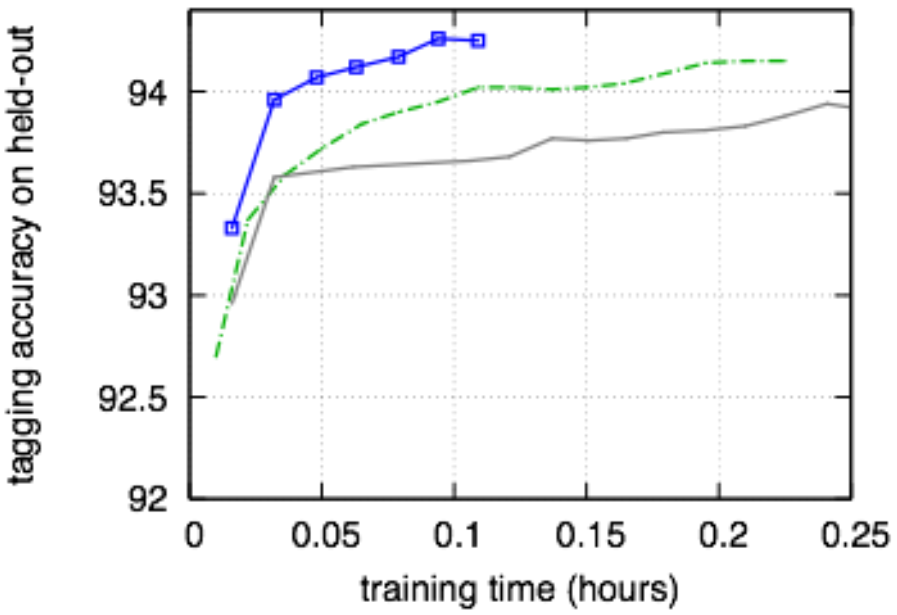
- Improves on the drawback of Early update
  - Slow learning: learns from only earliest mistake
- **Max-Violation fix**
  - Consider worst-mistake (maximum violation) instead of earliest-mistake for the weight update
  - More useful training data
  - Converges faster than early update

POS Tagging: Max-violation vs. Early vs. Standard

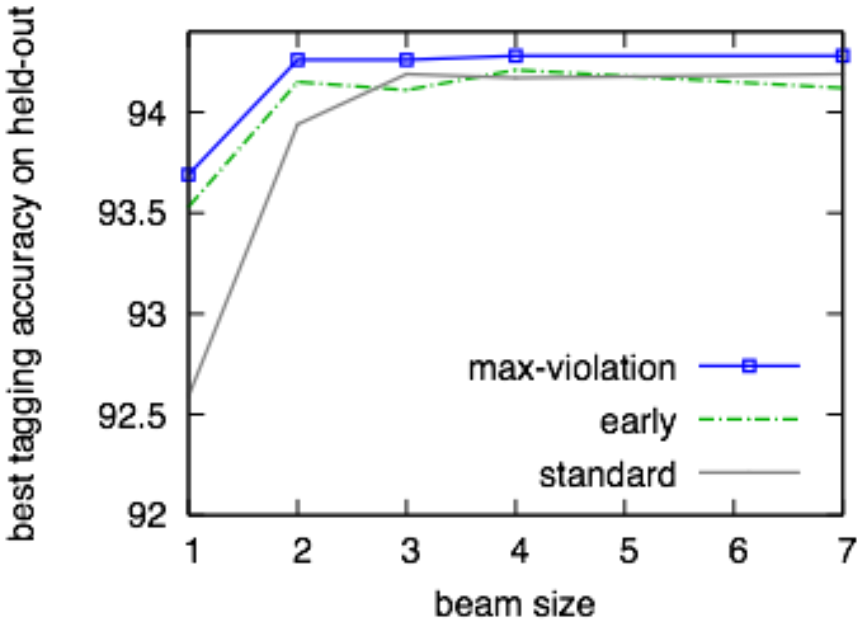
- **Early and Max-violation >> Standard at small beams**
  - Advantage shrinks as beam size increases
  - Max-violation converges faster than Early (and slightly better)

Source: Huang et al., 2012

Beam =2



Best accuracy vs. beam size

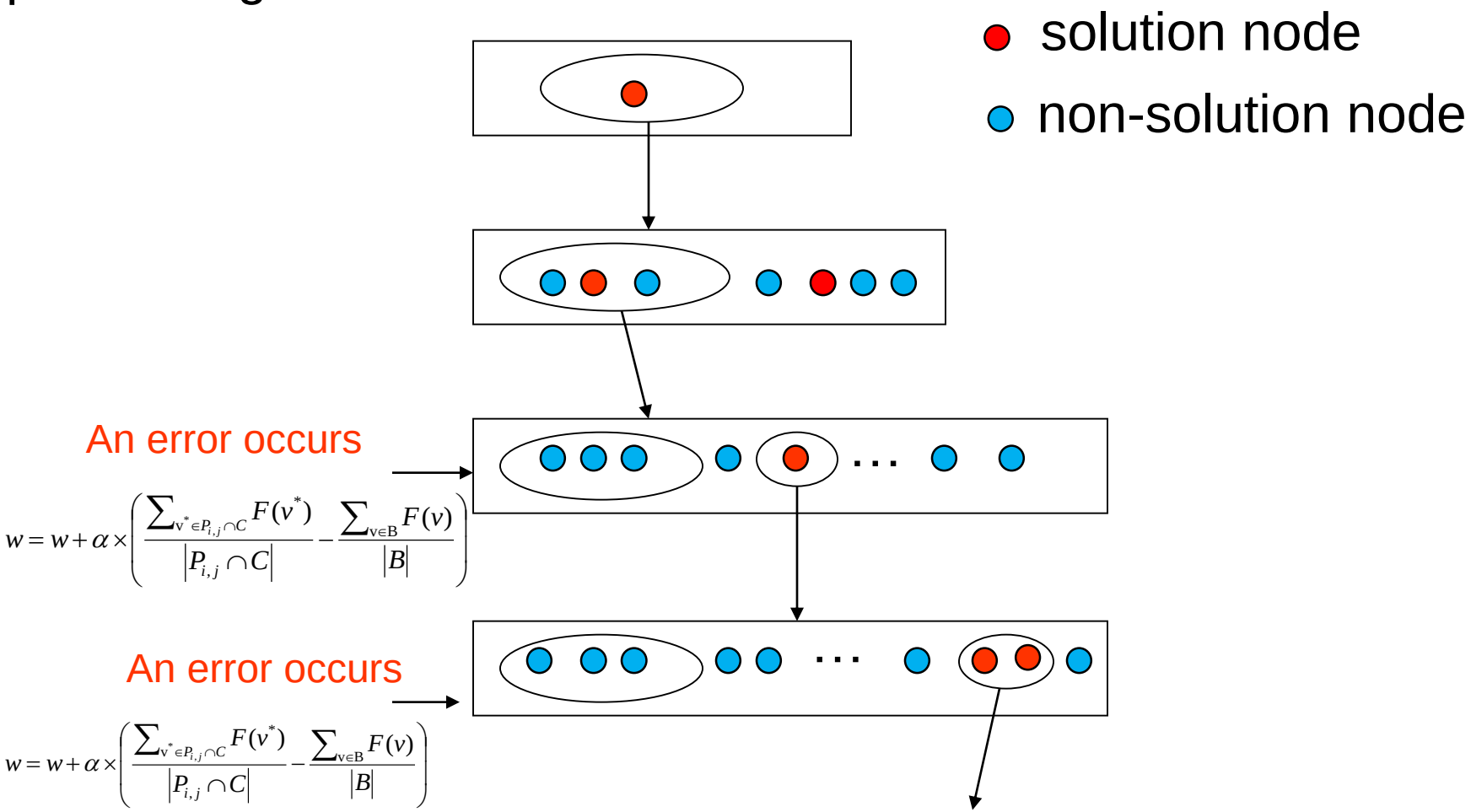


Beam Search Framework: LaSO

- **Search error:** NO target node in the beam
  - We cannot reach the goal node (correct structured output)
- **Weight update:** perceptron update
  - $w_{new} = w_{old} + \alpha \cdot (\psi_{avg}(target) - \psi_{avg}(non-target))$
  - $\psi_{avg}(target)$ = Average features of all target nodes in the candidate set
  - $\psi_{avg}(non-target)$= Average features of all non-target nodes in the candidate set
  - **Intuition:** increase the score of target nodes and decrease the score of the non-target nodes
- **Beam update:** reset beam with target nodes in the candidate set

LaSO Training: Illustration

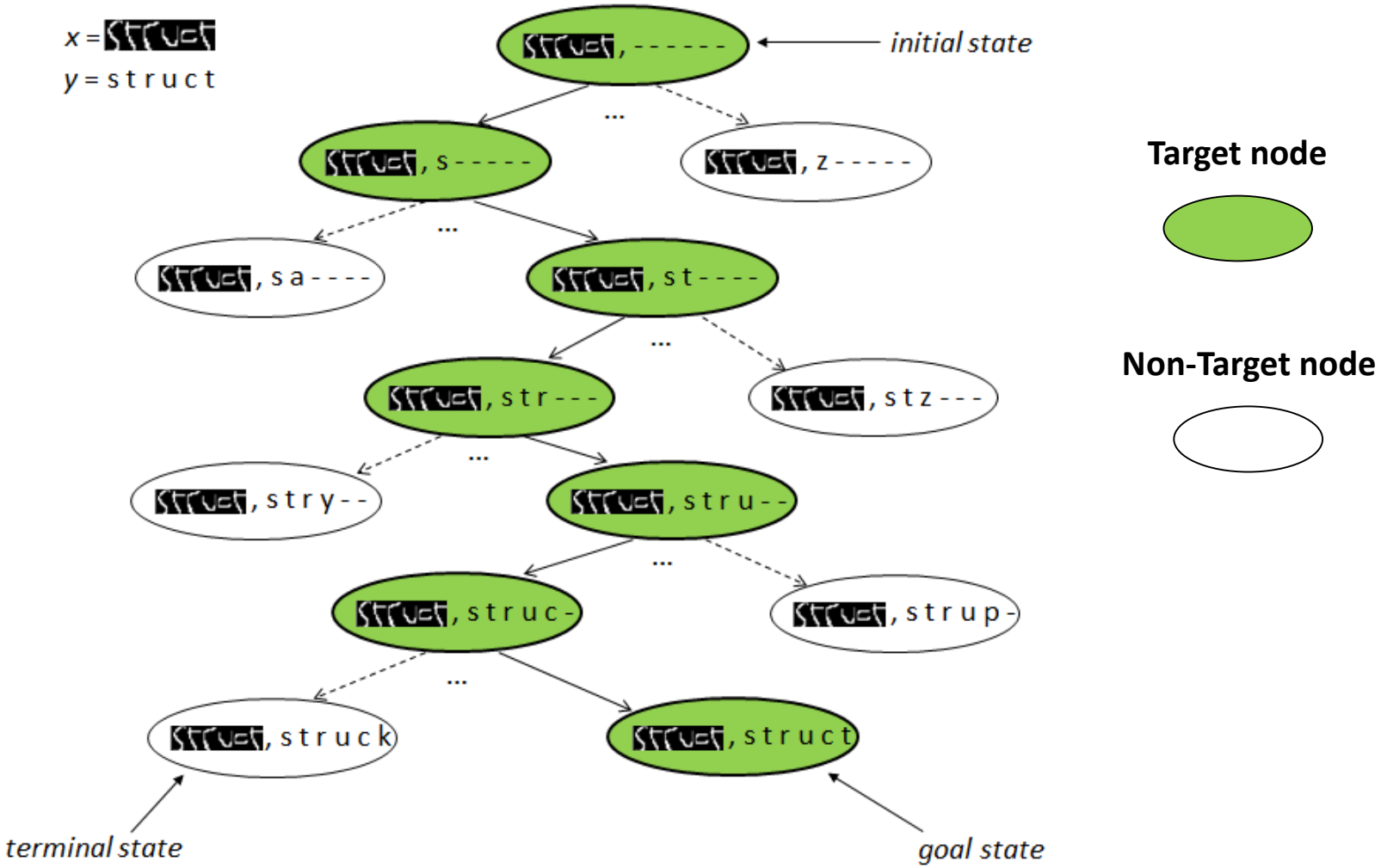
Basic Idea: repeatedly conduct search on training examples update weights when error occurs



LaSO Framework: Learning

- Represent heuristic function as a linear function
  - $H(n) = w \cdot \psi(n)$, where $\psi(n)$ stands for features of node n.
- **repeat**
  - For every training example (x, y)
  - Perform search with current weights
  - If search error , update weights
- **until** *convergence* or *max. iterations*

Search Error: Illustration



Search Error: Illustration for Beam Search

- **Search Error**
  - If the BEAM (internal memory of the search procedure – beam search) does NOT contain at least one target node
  - Why?
  - Because we cannot reach the goal node (correct complete output)
  - I call this definition **Conservative**

Search Error: Illustration for Beam Search

- **Search Error**
  - If the BEAM (internal memory of the search procedure – beam search) contains at least one non-target node ranked higher than a target node
  - I call this definition **Aggressive**
  - Does not exist in the literature!
- **Conservative vs. Aggressive**
  - **??**

Search Error: Illustration for Beam Search

- **Aggressive Error**
  - Seems over-constrained, which may make the learning problem harder than necessary?
  - More training data?
  - Increase the chances of reaching the terminal quickly?
- **Conservative Error**
  - relatively under-constrained. Good or Bad?
  - Less training data?
  - It can keep one target node in the beam and keep on expanding the non-target nodes – no progress?

Search Error: Illustration for Beam Search

- Checking for Search Error (w/ Beam Search)
  - Compute the candidate set
  - Compute the score of each node: $H(n) = w \cdot \psi(n)$
  - Sort the nodes based on the heuristic scores
  - Select the top scoring B nodes (BEAM)
  - Apply the search error definition on BEAM
- If search error , update weights

Weight Update: Illustration for Beam Search

- If search error, update weights

- **LaSO Weight Update:**
  - $w_{new} = w_{old} + \alpha \cdot (\psi_{avg}(target) - \psi_{avg}(non-target))$

$\psi_{avg}(target)$ = Average features of all target nodes in the candidate set

$\psi_{avg}(non-target)$ = Average features of all non-target nodes in the candidate set

  - **Intuition:** try to increase the score of target nodes and decrease the score of the non-target nodes

What happens after weight update?

- **Re-Score** the target nodes in the candidate set using the new weights
- **Reset the BEAM** with the top-scoring B target nodes
- Continue the search

Putting Everything Together: LaSO Training

- **Initialize** weights $w = 0$
- **repeat**
  - For every training example (x, y)
  - Perform search with current weights
  - If search error , update weights
  - Re-score the target nodes and reset BEAM
- **until** *convergence* or *max. iterations*

Putting Everything Together: LaSO Inference

- **Input:** learned weights w; structured input x

- **repeat**
  - Perform search with current weights
- **until** *reaching a terminal state*
- **Output:** the complete output y corresponding to the terminal state

Consistency, Convergence, and Hardness

- **Consistency Problem:**
  - Does a weight vector exist that can solve all the training problems correctly for a given search procedure?
- **Convergence:**
  - Under what definition(s) of margin will LaSO converge?
- **Hardness:**
  - Relating hardness of learning to the size of the beam
  - Intuitively, learning problem becomes easier as we increase the beam size
  - Degenerate case: BEAM = infinity, no learning is needed!

Additional Reading

- Yuehua Xu, Alan Fern, and Sungwook Yoon. (2009). **Learning Linear Ranking Functions for Beam Search with Application to Planning**. *Journal of Machine Learning Research (JMLR),*10, 1349-1388. http://www.jmlr.org/papers/volume10/xu09c/xu09c.pdf

- Beautiful paper! One of my favorite papers.