Structured Prediction

Kevin Duh

Intro to NLP, Fall 2019

Outline

- What is Structured Prediction; Why is it relevant to NLP?
- Generative vs. Discriminative; Local vs. Global
- Models for sequence labeling
  - HMM, MEMM
  - CRF, Structure Perceptron, Structured SVM

This lecture ties together many of the concepts we've seen this semester!

Machine Learning Abstractions

- Training data
  - Input: **x** / Output: **y**
  - Lots of $\{(\mathbf{x}_i, \mathbf{y}_i)\}$ **i**=1,2,…,N
- Goal: build model F(**x**) on training data, generalize to test data: $\mathbf{y}_{\mathbf{prediction}} = F(\mathbf{x}_{\mathbf{test}})$, $\mathbf{y}_{\mathbf{prediction}}$ vs $\mathbf{y}_{\mathbf{truth}}$
- What is the structure of **x**? What is the structure of **y**?
  - changes the model from the machine learning perspective

Machine Learning Abstractions

- Standard setup in machine learning:
  - **x** is a vector in R^D
  - **y** is a label from {class1, class2, class3, … classK}
- Characteristics of NLP problems:
  - **x** is a word or sentence: discrete input
  - **y** has large output space

Structured Output Example: Variable-Length Sequences

- **Input: Image**

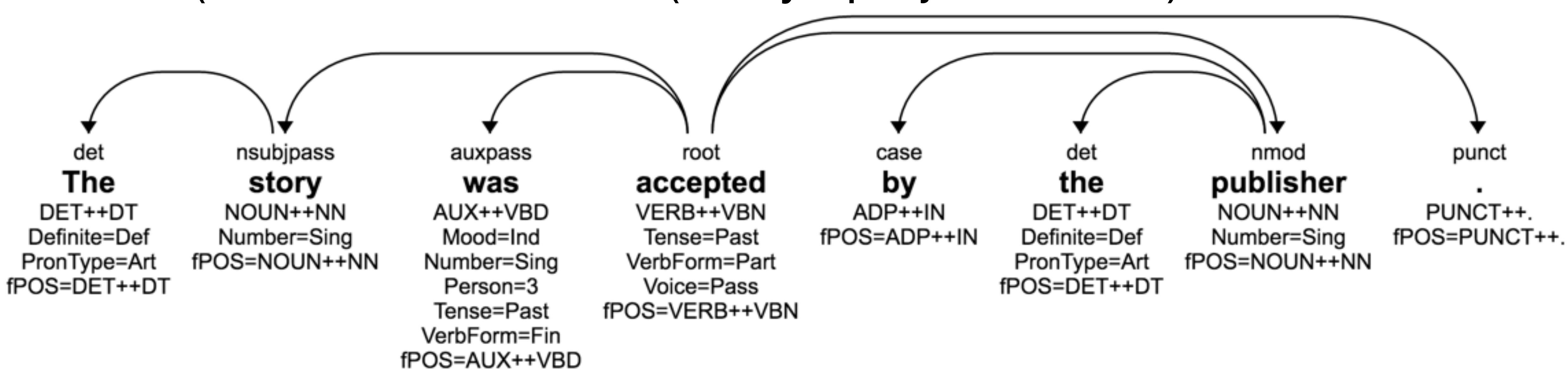
Caption text generation output space:
{ all possible English sentences }

a cute dog
a very cute dog
super cute puppy
adorable puppy looking at me
....

Image recognition output label space:
{ cat, dog, door, nose, bug, ... }

Structured Output Example: Trees

- **<u>Input:</u>**
  - Sentence: The story was accepted by the publisher .
- **<u>Output: Depedency tree</u>**
  - Still N labels (one head per word), but has constraints (must be a valid tree (mabye projective tree)



The size of output space

- The size of the output space depends on the problem
- For text generation problems:
  - Assume vocabulary size V and max length L
  - Space: $V + V \times V + \ldots V \times V \times V + \ldots V^L$
  - Sometimes cannot assume max length, use <stop> symbol
- For non-generation problems:
  - Space could be polynomial or exponential, but has structure that can be exploited

What is Structured Prediction

- Definition:
  - A ML problem with a large output space that contains dependencies (structure) between variables
  - Additionally, sometimes the desired loss function does not decompose well between these variables
- Very prevelant in NLP!

Outline

- What is Structured Prediction; Why is it relevant to NLP?
- Generative vs. Discriminative; Local vs. Global
- Models for sequence labeling
    - HMM, MEMM
    - CRF, Structure Perceptron, Structured SVM

Generative vs Discriminative Models

- Input **x**, Output **y**
- Generative model defines p(**x**,**y**)
  - If we condition on **y**, we can generate samples **x**
  - We can still compute p(**y**|**x**) = p(**x**,**y**)/p(**x**) and do prediction
- Discriminative model defines p(**y**|**x**)
  - Directly describes quantity we care about for prediction
- (Note: terminology is not always consistent in the research literature. Possible to have p(**x**,**y**) but trained discriminatively)

Local vs. Global Models

- Input **x**, Output **y**
  - Let's say **y** is a sequence of N labels (y_1, y_2, .. y_N)
- Local models treat each of the N predictions as separate
  - Totally independent: $p(y_1|\mathbf{x})$, $p(y_1|\mathbf{x})$, $p(y_3|\mathbf{x})$
  - Add dependency (greedy): $p(y_1|\mathbf{x})$, $p(y_2|y_1,\mathbf{x})$, $p(y_3|y_2,y_1,\mathbf{x})$
- Global models treat N predictions as one joint decision

Example for Sequence Labeling

	Generative	Discriminative
Local		MEMM
Global	HMM	CRF Structured Perceptron Structured SVM

Outline

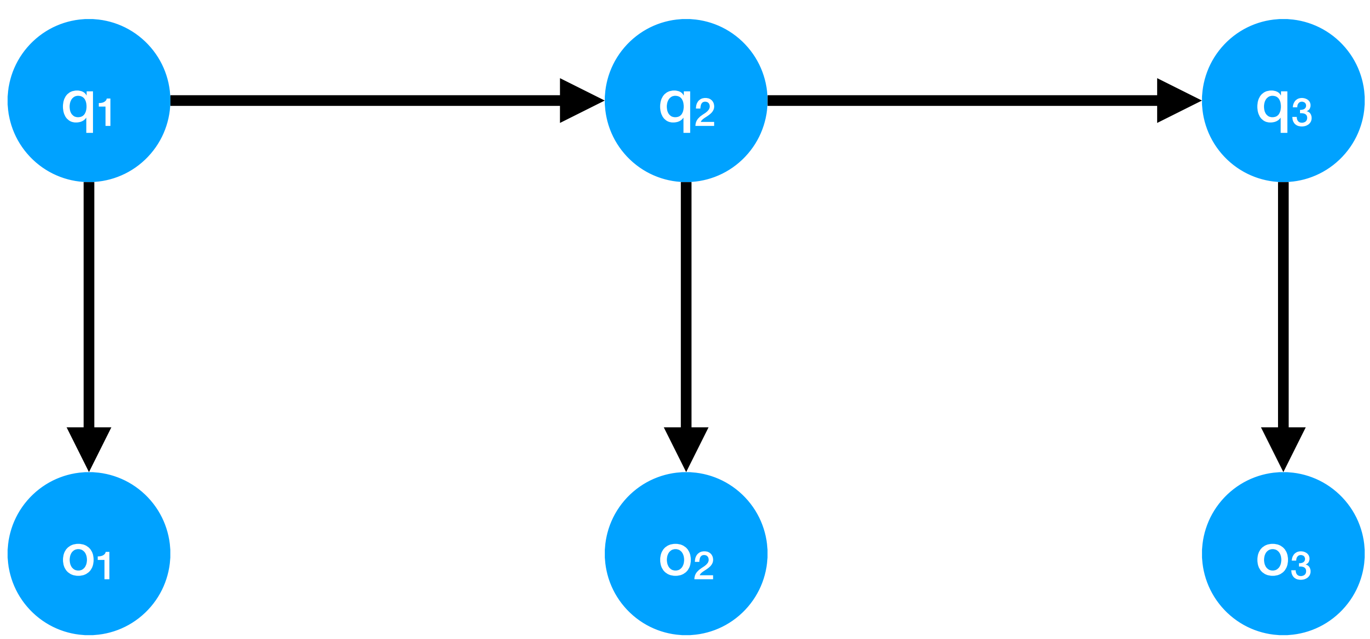
- What is Structured Prediction; Why is it relevant to NLP?
- Generative vs. Discriminative; Local vs. Global
- Models for sequence labeling
  - HMM, MEMM
  - CRF, Structure Perceptron, Structured SVM

Hidden Markov Models (HMM)

Generative

$$P(O, Q) = P(O|Q)P(Q) = \prod_{t=1}^{T} P(o_t|q_t) \times \prod_{t=1}^{T} P(q_t|q_{t-1})$$

Global

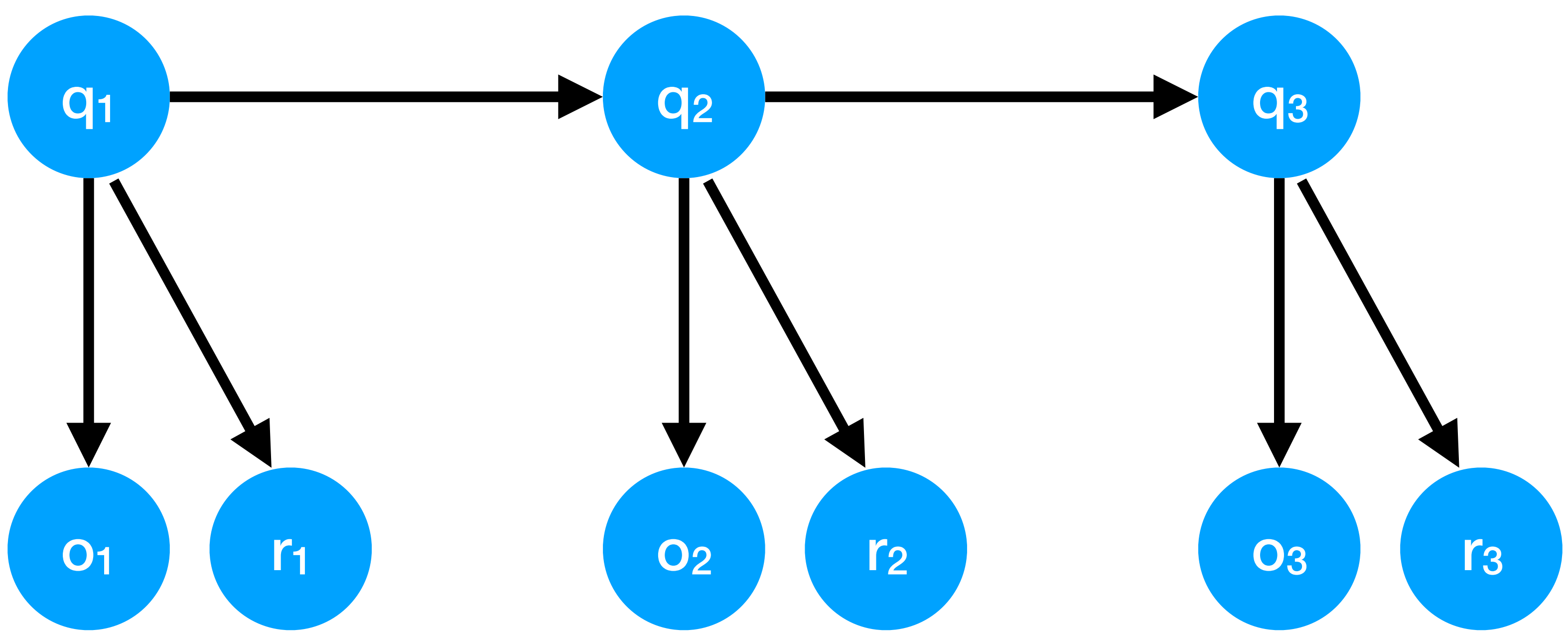


Generative Model Demerit: Difficult to add arbitrary features

Suppose I want to incorporate many features
These all need to be "generated"

$$P(O,Q) = P(O|Q)P(Q) = \prod_{t=1}^{T} P(o_t|q_t) \times \prod_{t=1}^{T} P(q_t|q_{t-1})$$

$$P(O,R,Q) = P(O,R|Q)P(Q) = \prod_{t=1}^{T} P(r_t|q_t) \times \prod_{t=1}^{T} P(o_t|q_t) \times \prod_{t=1}^{T} P(q_t|q_{t-1})$$

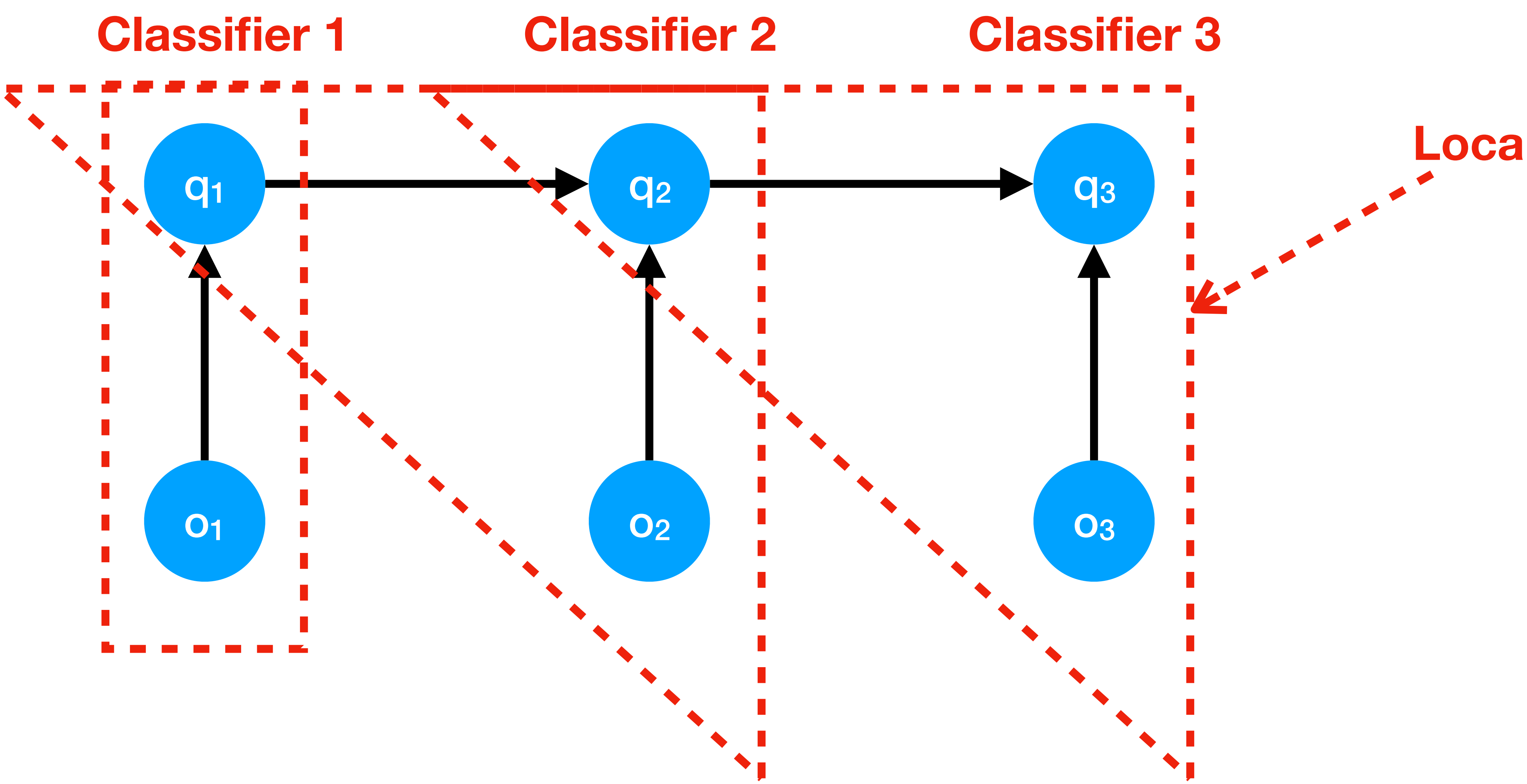


But need to be careful about "feature selection", otherwise waste modeling power on features that don't matter for classification. e.g. imagine r_t is random or redundant. (model assumes feature independence)

Maximum Entropy Markov Models (MEMM)

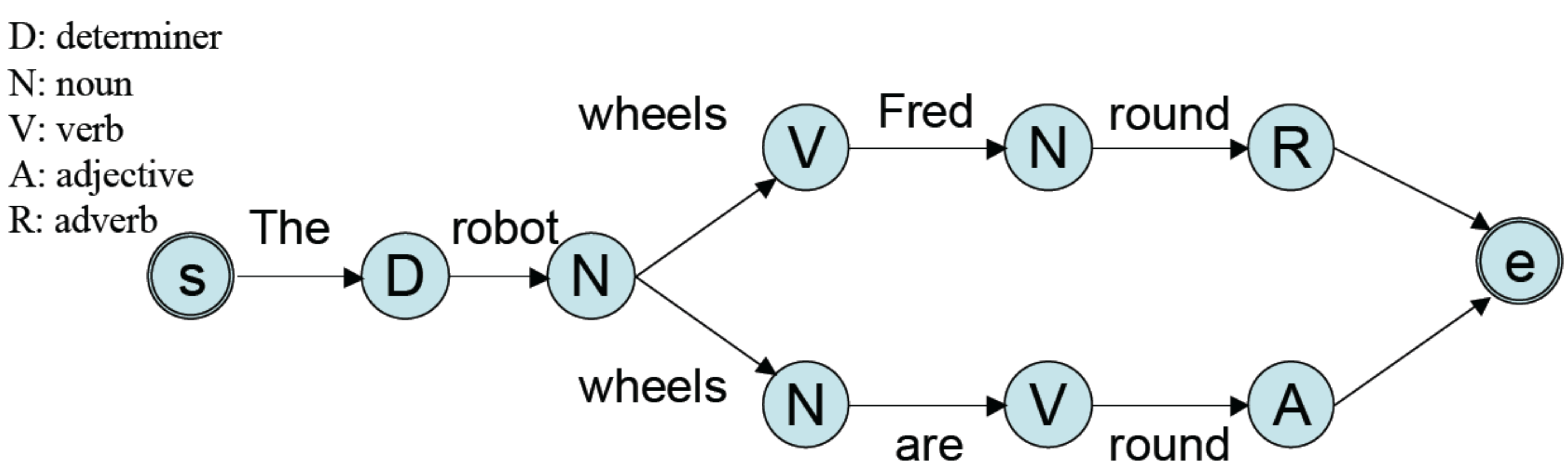
Discriminative: log-linear models

$$P(q_1|o_1) \propto \exp(\sum_k \theta_k \cdot f_k(q_1, o_1)) \qquad P(q_t|o_t, q_{t-1}) \propto \exp(\sum_k \theta_k \cdot f_k(q_t, o_t, q_{t-1}))$$



Local Model Demerit: Label Bias

- POS tagging example
- Observation: The robot wheels are round



Due to per-state normalization: if P(V|N,wheels) > P(N|N,wheels), MEMM stuck in upper path regardless of observation

Example from Wallach (2002). Efficient Training of Conditional Random Fields. M. Sc. thesis, Univ. of Edinburgh

Label Bias Problem

- The problem: States with low-entropy next-state distributions tend to ignore observations
  - due to per-state normalization, i.e. transitions leaving a state only compete against each other
- Solution:
  - need global model that accounts for whole sequence
  - amplify/dampen probability at individual transitions: finite-state model with un-normalized transition probability

Outline

- What is Structured Prediction; Why is it relevant to NLP?
- Generative vs. Discriminative; Local vs. Global
- Models for sequence labeling
  - HMM, MEMM
  - CRF, Structure Perceptron, Structured SVM

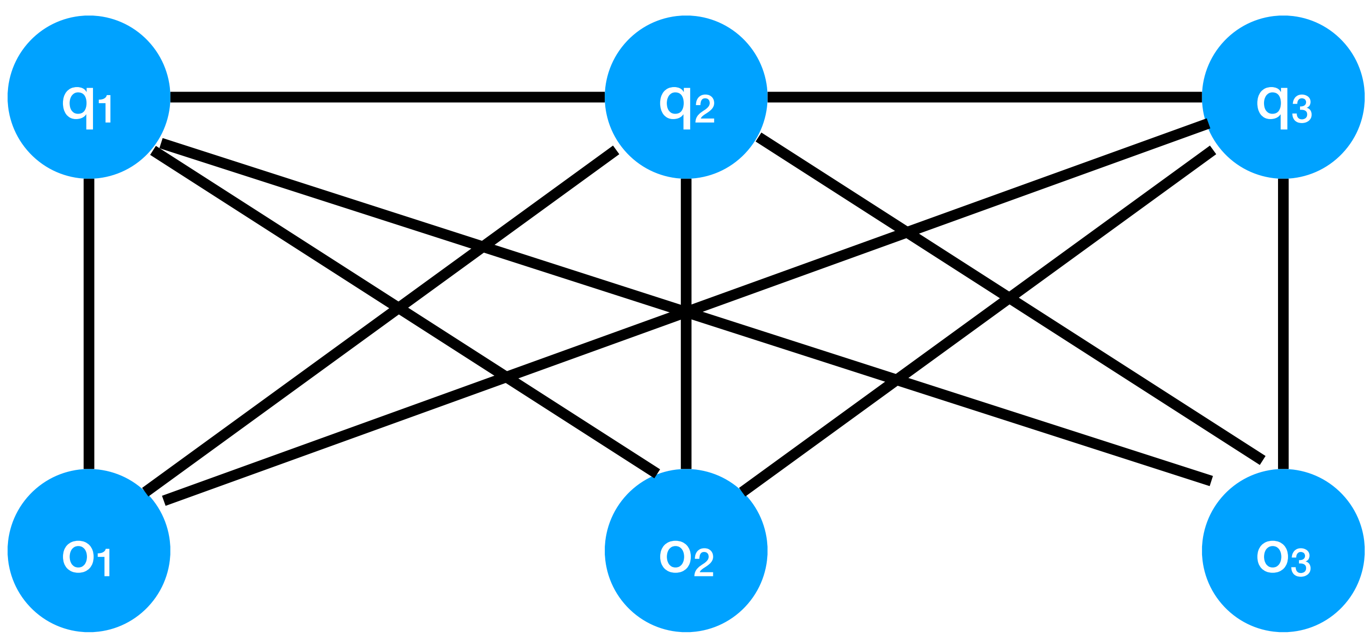
Intuition: use log-linear model like MEMM, but have global normalization

- Define distribution over all possible sequences of Q, conditioned on O
  - (may be intractable depending on assumptions)

$$P(Q|O) = P(q_1, q_2, \ldots, q_N | o_1, o_2, \ldots, o_N)$$
$$\propto \exp(\sum_k \theta_k \cdot f_k(q_1, q_2, \ldots, q_N, o_1, o_2, \ldots, o_N))$$

Linear-Chain Conditional Random Field (CRF)

$$P(Q|O) \propto \exp(\sum_{i,k} \theta_k \cdot f_k(q_i, q_{i-1}, O) + \sum_{i,j} \theta_j \cdot f_j(q_i, O))$$



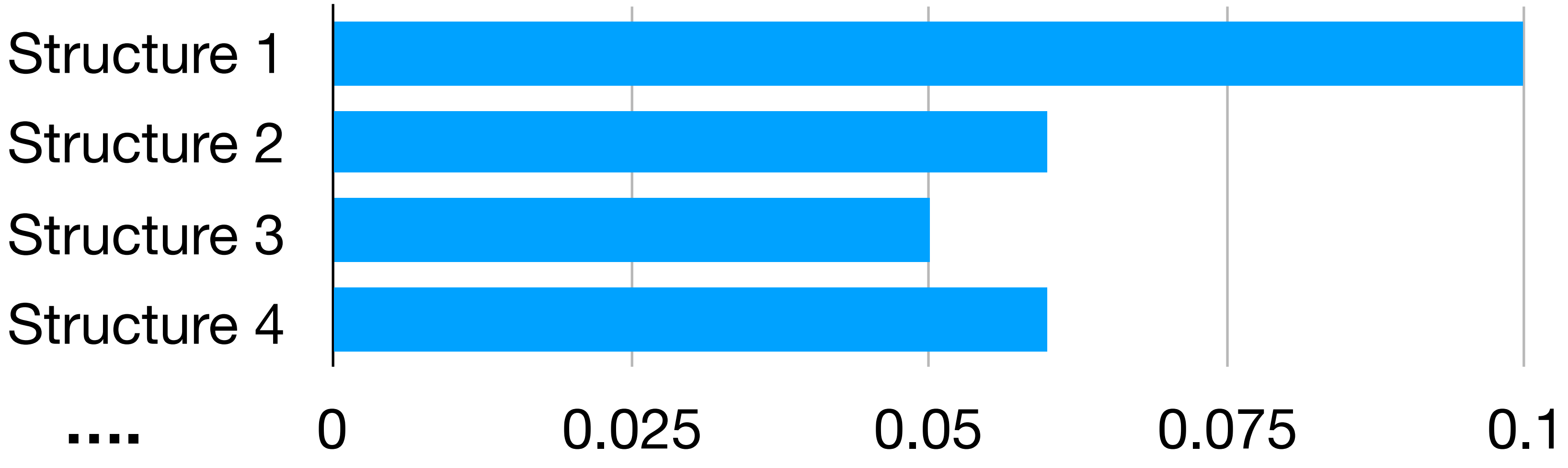
Training is similar to what we derived for log-linear models, but need efficient inference (Dynamic Programming) to compute partition function over all sequences

General CRF

- Cliques c define variables that should interact

$$P(Q|O) = \frac{\exp(\sum_{c,k} \theta_k \cdot f_k(c, Q^{(c)}, O))}{\sum_{Q'} \exp(\sum_{c,k} \theta_k \cdot f_k(c, Q'^{(c)}, O))}$$

- Distribution over all possible output structures



What if we don't need a probabilistic model?

$$P(Q|O) = \frac{\exp(\sum_{c,k} \theta_k \cdot f_k(c, Q^{(c)}, O))}{\sum_{Q'} \exp(\sum_{c,k} \theta_k \cdot f_k(c, Q'^{(c)}, O))}$$

- We only need to output a single "best" Q given O

$$S(Q|O) = \sum_{c,k} \theta_k \cdot f_k(c, Q^{(c)}, O)$$

$$\hat{Q} = \arg\max S(Q|O) = \arg\max \sum_{c,k} \theta_k \cdot f_k(c, Q^{(c)}, O)$$

Structured Perceptron

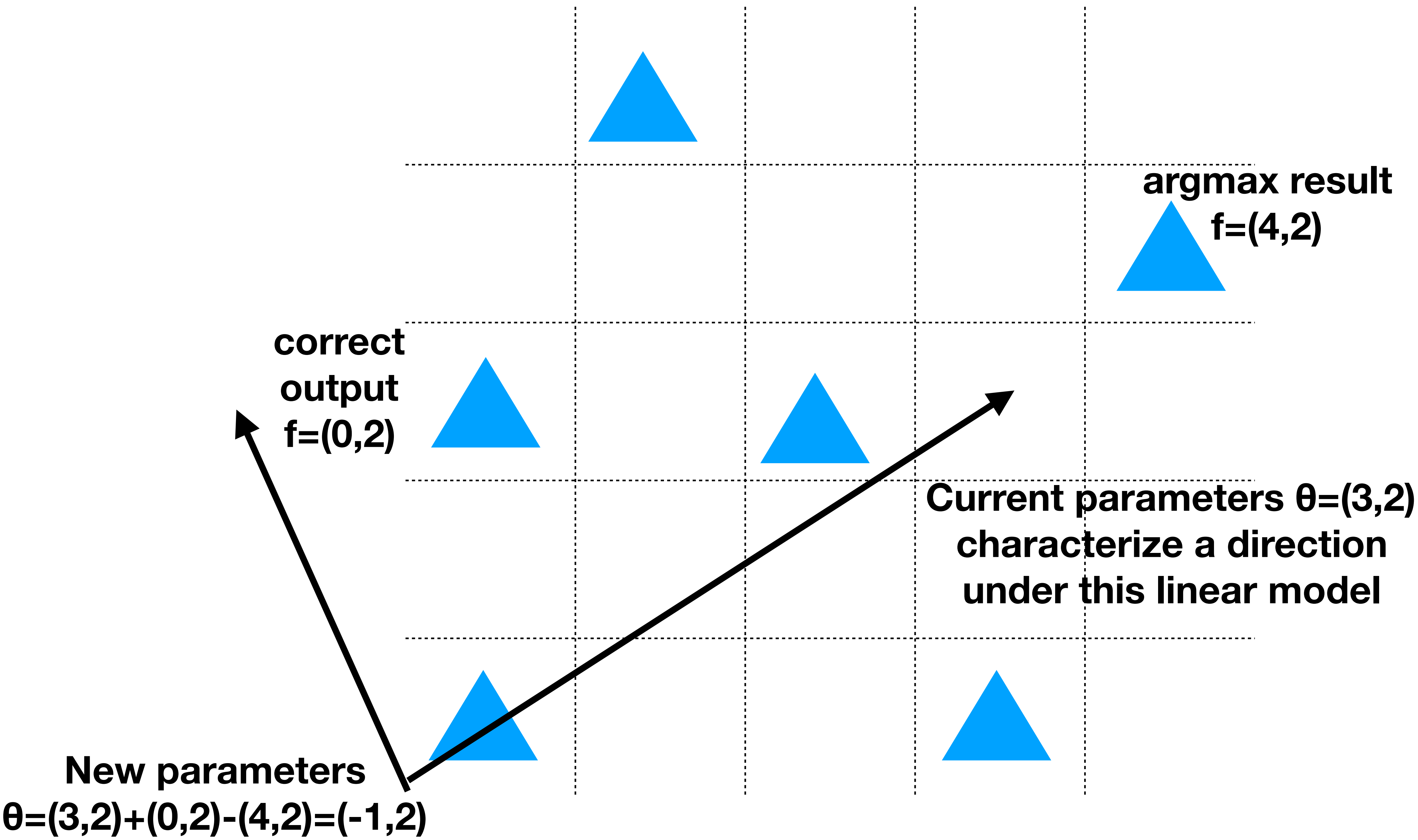
- Define features over structure: $$\sum_k \theta_k \cdot f_k(Q, O)$$
- Training procedure:
  - While not converged:
    - Draw training sample (Q, O)
    - Decode: $$\hat{Q} = \arg \max_{Q' \in G(O)} \sum_k \theta_k \cdot f_k(Q', O)$$
    - If incorrect $Q \neq \hat{Q}$; update

$$\theta_k \mathrel{+}= f_k(Q, O) - f_k(\hat{Q}, O)$$

G(O) denotes all output structure of O. Only requirement is a decoder that can search over this G(O)

Add positive example, subtract negative example

Structure Perceptron: Geometric View



Structured Perceptron for HMM

$$P(O,Q) = \prod_{t=1}^{T} P(o_t|q_t) \times P(q_t|q_{t-1})$$

$$\log P(O,Q) = \sum_{t=1}^{T} \log P(o_t|q_t) + \log P(q_t|q_{t-1})$$

$$= \sum_{s} \log P(o_t|q_t = s) Count(s)$$

$$+ \sum_{s,s'} \log P(q_t = s|q_{t-1} = s') Count(s,s')$$

Weights θ

Features f

Structured Perceptron vs. CRF

- If we use SGD update for CRF, then the update turns out very similar (modulo regularization, learning rate, etc.)

- Structured Perceptron

$$\theta_k \mathrel{+}= f_k(Q, O) - f_k(\hat{Q}, O)$$

Argmax over all output structures

- CRF

$$\theta_k \mathrel{+}= f_k(Q, O) - E_Q[f_k(Q, O)]$$

Expectation over all output structures

Margin

- Our structured perceptron implements:
  - Score(correct structure) ≥ Score(any other structure)
- We can make this more robust by adding a margin:
  - Score(correct structure) ≥ Score(any other struct) + Positive constant
- Further, we can incorporate domain knowledge:
  - Score(correct structure) ≥ Score(very bad structure) + Large constant
  - Score(correct structure) ≥ Score(not bad structure) + Small constant

Structured Support Vector Machine (Large-Margin Structured Classifier)

- We desire scores such that these constraints are satisfied

$$\theta^T f(Q, O) \geq \theta^T f(Q', O) + l(Q, Q') \quad \forall Q'$$

- Rather than enumerating all constraints, we only need the max:

$$\theta^T f(Q, O) \geq \max_{Q'} [\theta^T f(Q', O) + l(Q, Q')]$$

- Update similar to structured perceptron, but different negative example:

$$\theta_k \mathrel{+}= f_k(Q, O) - f_k(Q^*, O)$$

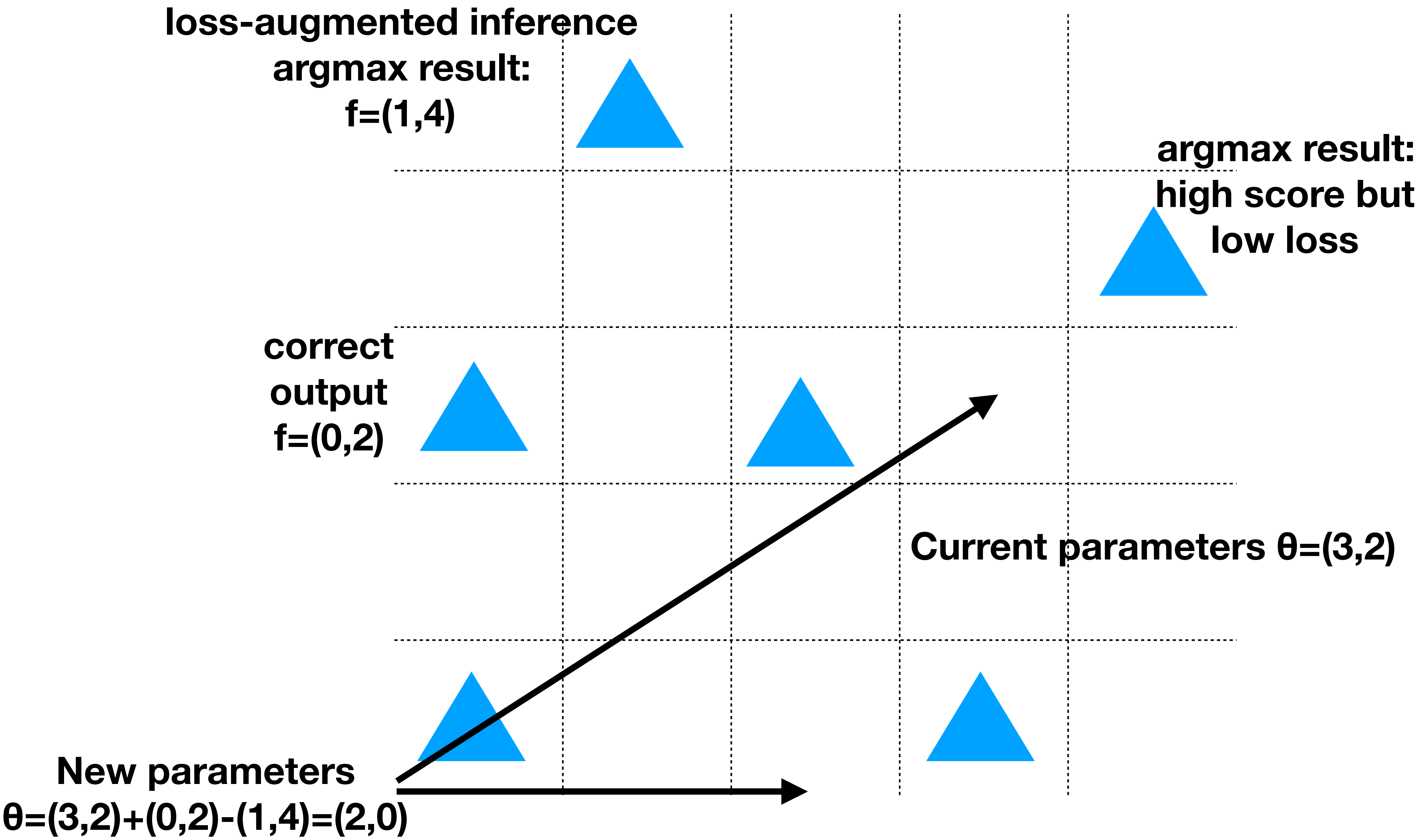
$$Q^* = \arg\max_{Q'} [\theta^T f(Q', O) + l(Q, Q')]$$

Loss-augmented inference: assumes your decoder can exploit structure in l(Q,Q')

Structure SVM: Geometric View

$$\theta^T f(Q,O) \geq \theta^T f(Q',O) + l(Q,Q') \quad \forall Q'$$

score loss/penalty



Big picture: Structured Perceptron/SVM

- Simple learning procedure. All you need is a decoder
- Discriminative (allows arbitrary features) and Global (considers all decisions jointly)
- Caveat: Decoder has to search over all large output space. Often feature definition affects tractability

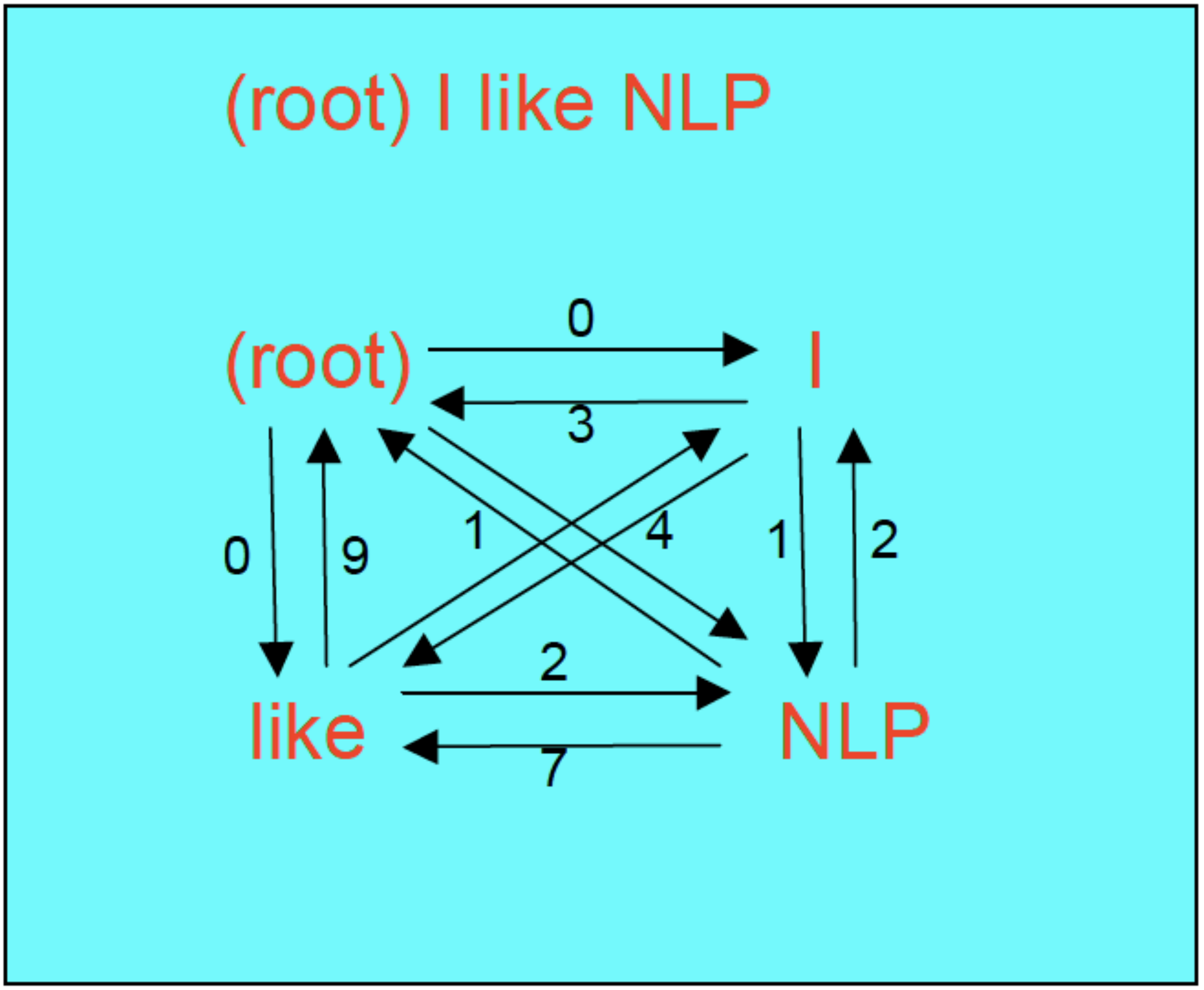
Another example: Dependency Parsing with Maximum Spanning Trees

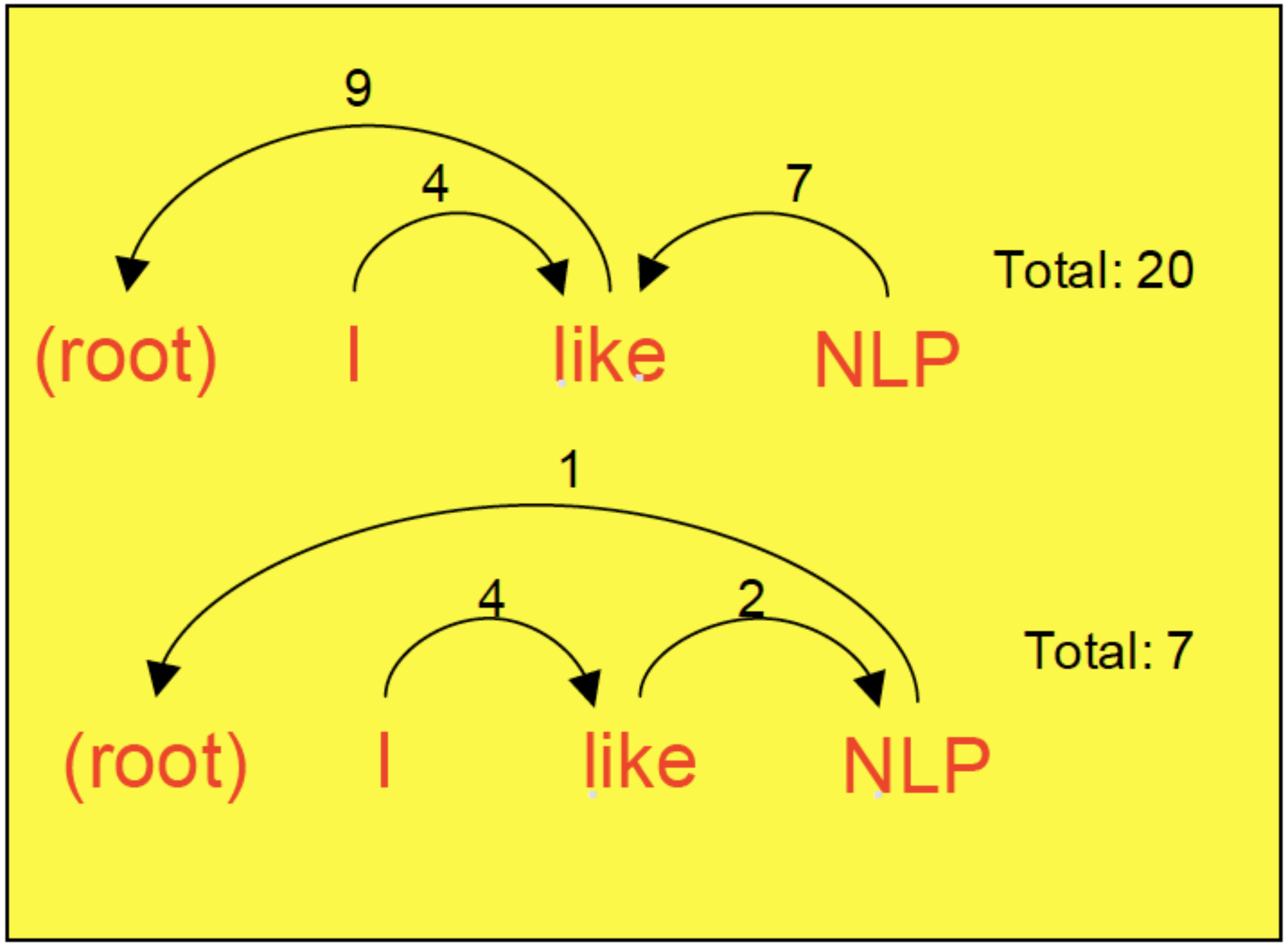
- Define the score of a dependency parse as the sum of all edge scores

$$predictedtree = \arg \max_{all\ trees} \sum_{edge \in tree} edgescore(i,j)$$

$$= \arg \max_{all\ trees} \sum_{edge \in tree} \sum_{k} \theta_k f_k(i,j)$$

- Argmax can be computed by maximum spanning tree algorithm


(root) I like NLP
(root)
I
like
NLP
0
3
0
9
1
4
1
2
2
7


9
4
7
Total: 20
(root)
I
like
NLP
1
4
2
Total: 7
(root)
I
like
NLP

Summary

	Generative	Discriminative
Local		**MEMM**: Label bias problem Note: Many Recurrent Neural Net models have label bias too
Global	**HMM**: Cannot incorporate arbitrary features	**CRF**: Extension of log-linear model to structured output space **Structured Perceptron**: Just need a decoder. My 1st bet **Structured SVM**: Incorporates concept of margin

Recurring theme: efficient computation that exploits structure. This is where domain knowledge helps!