COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

Sequence Labeling II

CMSC 470

Marine Carpuat

Recap: We know how to perform POS tagging with structured perceptron

- An example of sequence labeling tasks
- Requires a predefined set of POS tags
  - Penn Treebank commonly used for English
  - Encodes some distinctions and not others
- Given annotated examples, we can address sequence labeling with multiclass perceptron
  - but computing the argmax naively is expensive
  - constraints on the feature definition make efficient algorithms possible

We can view POS tagging as classification and use the perceptron again!

$$\hat{y} = \text{argmax}_{\hat{y} \in \mathcal{Y}(x)} \, w \cdot \phi(x, \hat{y})$$

Algorithm 40 StructuredPerceptronTrain(**D**, *MaxIter*)

$w \leftarrow 0$ // initialize weights
for *iter* = 1 … *MaxIter* **do**
  for all $(x,y) \in$ **D** **do**
    $\hat{y} \leftarrow \text{argmax}_{\hat{y} \in \mathcal{Y}(x)} \, w \cdot \phi(x, \hat{y})$ // compute prediction
    if $\hat{y} \neq y$ **then**
      $w \leftarrow w + \phi(x, y) - \phi(x, \hat{y})$ // update weights
    end if
  end for
end for
return w // return learned weights

Algorithm from CIML chapter 17

Feature functions for sequence labeling

$x =$ " monsters eat tasty bunnies "

$y =$ noun verb adj noun

- Standard features of POS tagging
  - **Unary features:** capture relationship between input x and a **single label** in the output sequence y
    - e.g., "# times word w has been labeled with tag l for all words w and all tags l"
  - **Markov features:** capture relationship between **adjacent labels** in the output sequence y
    - e.g., "# times tag l is adjacent to tag l' in output for all tags l and l'"
- Given these feature types, the size of the feature vector is constant with respect to input length

Example from CIML chapter 17

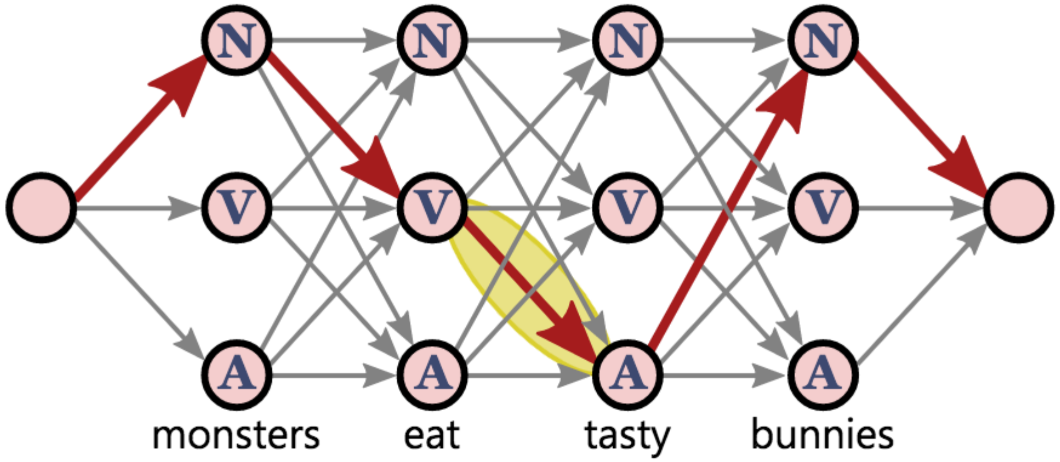
Decomposability

- If **features decompose over the input sequence**, then we can decompose the perceptron score as follows

$$w \cdot \phi(x, y) = w \cdot \sum_{l=1}^{L} \phi_l(x, y)$$
$$= \sum_{l=1}^{L} w \cdot \phi_l(x, y)$$

- This holds for unary and Markov features

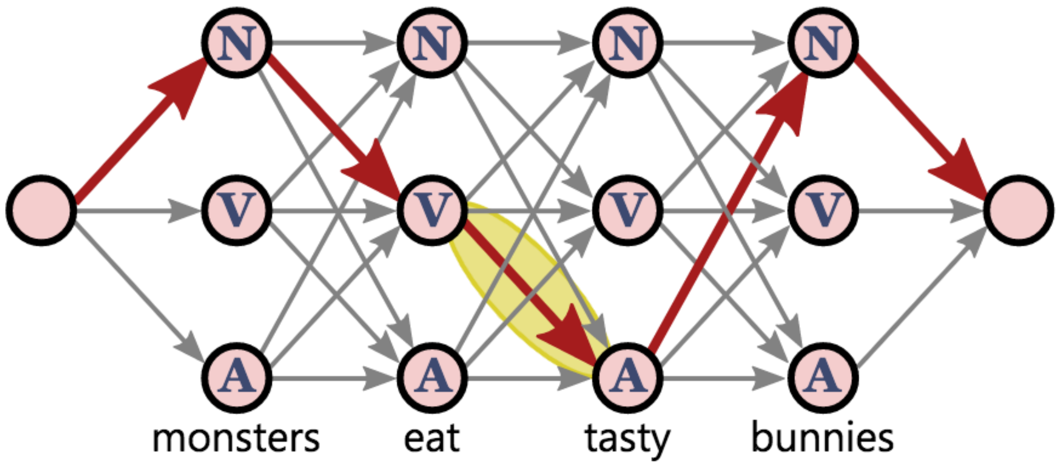
Solving the argmax problem for sequences efficiently with dynamic programming



- Possible when features decompose over input

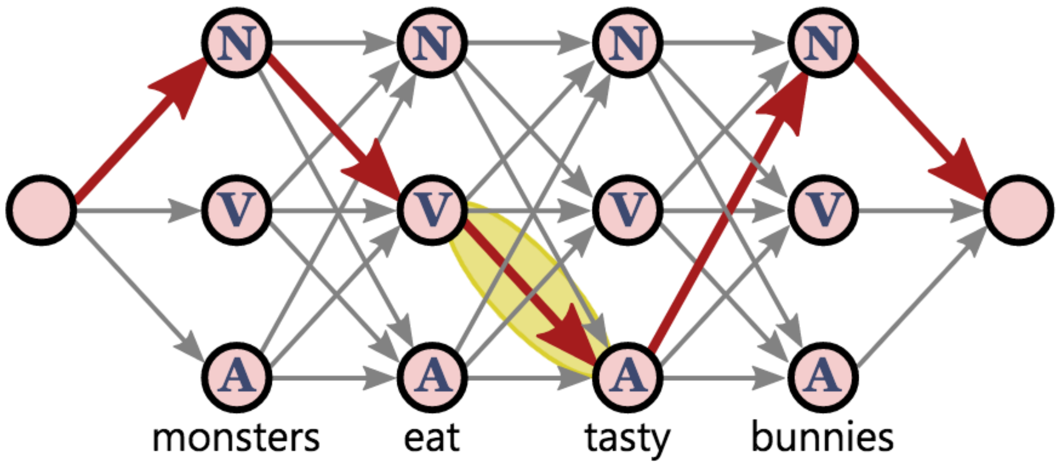
- We can represent the search space as a trellis/lattice
  - Any path represents a labeling of input sentence
  - Each edge receives a weight such that adding weights along the path corresponds to score for input/ouput configuration

Defining the Viterbi lattice for our POS tagger
(assuming features from slide 4)



- Each node corresponds to one time step (or position in the input sequence) and one POS tag
- Each edge in the lattice connects from time *l* to *l+1*, and from tag *k'* to *k*

Defining the Viterbi lattice for our POS tagger
(assuming features from slide 4)



- When features decompose over input, we can

  - Define the score of the best path in lattice up to and including position *l* that labels the *l-th* word as *k*

$$\alpha_{l,k} = \max_{\hat{y}_{1:l-1}} \boldsymbol{w} \cdot \phi_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)$$

  - And compute this score recursively

$$\alpha_{l+1,k} \leftarrow \max_{k'} \left[\alpha_{l,k'} + \boldsymbol{w} \cdot \phi_{l+1}(\boldsymbol{x}, \langle \ldots, k', k \rangle)\right]$$

Best prefix up to l ending in k'

Score contribution of adding k to prefix

Deriving the recursion

$$\alpha_{0,k} = 0 \quad \forall k$$

$$\zeta_{0,k} = \emptyset \quad \forall k$$

the score for any empty sequence is zero

$$\alpha_{l+1,k} = \max_{\hat{\boldsymbol{y}}_{1:l}} \boldsymbol{w} \cdot \phi_{1:l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)$$

Deriving the recursion

$$\alpha_{0,k} = 0 \quad \forall k$$

$$\zeta_{0,k} = \emptyset \quad \forall k$$

the score for any empty sequence is zero

$$\alpha_{l+1,k} = \max_{\hat{y}_{1:l}} \boldsymbol{w} \cdot \phi_{1:l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)$$

separate score of prefix from score of position l+1

$$= \max_{\hat{y}_{1:l}} \boldsymbol{w} \cdot \Big(\phi_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}}) + \phi_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)\Big)$$

Deriving the recursion

$$\alpha_{0,k} = 0 \quad \forall k$$

$$\zeta_{0,k} = \emptyset \quad \forall k$$

the score for any empty sequence is zero

$$\alpha_{l+1,k} = \max_{\hat{y}_{1:l}} \boldsymbol{w} \cdot \phi_{1:l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)$$

separate score of prefix from score of position l+1

$$= \max_{\hat{y}_{1:l}} \boldsymbol{w} \cdot \left(\phi_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}}) + \phi_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)\right)$$

distributive law over dot products

$$= \max_{\hat{y}_{1:l}} \left[\boldsymbol{w} \cdot \phi_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}}) + \boldsymbol{w} \cdot \phi_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)\right]$$

Deriving the recursion

$$\alpha_{0,k} = 0 \quad \forall k$$

$$\zeta_{0,k} = \varnothing \quad \forall k$$

the score for any empty sequence is zero

$$\alpha_{l+1,k} = \max_{\hat{y}_{1:l}} \boldsymbol{w} \cdot \phi_{1:l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)$$

separate score of prefix from score of position l+1

$$= \max_{\hat{y}_{1:l}} \boldsymbol{w} \cdot \Big(\phi_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}}) + \phi_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)\Big)$$

distributive law over dot products

$$= \max_{\hat{y}_{1:l}} \Big[\boldsymbol{w} \cdot \phi_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}}) + \boldsymbol{w} \cdot \phi_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)\Big]$$

separate out final label from prefix, call it k’

$$= \max_{\hat{y}_{1:l-1}} \max_{k'} \Big[\boldsymbol{w} \cdot \phi_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k') + \boldsymbol{w} \cdot \phi_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k' \circ k)\Big]$$

Deriving the recursion

$$\alpha_{0,k} = 0 \quad \forall k$$

$$\zeta_{0,k} = \emptyset \quad \forall k$$

the score for any empty sequence is zero

$$\alpha_{l+1,k} = \max_{\hat{y}_{1:l}} \boldsymbol{w} \cdot \phi_{1:l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)$$

separate score of prefix from score of position l+1

$$= \max_{\hat{y}_{1:l}} \boldsymbol{w} \cdot \Big(\phi_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}}) + \phi_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)\Big)$$

distributive law over dot products

$$= \max_{\hat{y}_{1:l}} \Big[\boldsymbol{w} \cdot \phi_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}}) + \boldsymbol{w} \cdot \phi_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)\Big]$$

separate out final label from prefix, call it k'

$$= \max_{\hat{y}_{1:l-1}} \max_{k'} \Big[\boldsymbol{w} \cdot \phi_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k') + \boldsymbol{w} \cdot \phi_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k' \circ k)\Big]$$

swap order of maxes, and last term doesn't depend on prefix

$$= \max_{k'} \Big[\Big[\max_{\hat{y}_{1:l-1}} \boldsymbol{w} \cdot \phi_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k')\Big] + \boldsymbol{w} \cdot \phi_{l+1}(\boldsymbol{x}, \langle \dots, k', k \rangle)\Big]$$

Deriving the recursion

$$\alpha_{0,k} = 0 \quad \forall k$$

$$\zeta_{0,k} = \varnothing \quad \forall k$$

the score for any empty sequence is zero

$$\alpha_{l+1,k} = \max_{\hat{y}_{1:l}} \boldsymbol{w} \cdot \phi_{1:l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)$$

separate score of prefix from score of position l+1

$$= \max_{\hat{y}_{1:l}} \boldsymbol{w} \cdot \Big(\phi_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}}) + \phi_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)\Big)$$

distributive law over dot products

$$= \max_{\hat{y}_{1:l}} \Big[\boldsymbol{w} \cdot \phi_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}}) + \boldsymbol{w} \cdot \phi_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k)\Big]$$

separate out final label from prefix, call it k'

$$= \max_{\hat{y}_{1:l-1}} \max_{k'} \Big[\boldsymbol{w} \cdot \phi_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k') + \boldsymbol{w} \cdot \phi_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k' \circ k)\Big]$$

swap order of maxes, and last term doesn't depend on prefix

$$= \max_{k'} \Big[\Big[\max_{\hat{y}_{1:l-1}} \boldsymbol{w} \cdot \phi_{1:l}(\boldsymbol{x}, \hat{\boldsymbol{y}} \circ k')\Big] + \boldsymbol{w} \cdot \phi_{l+1}(\boldsymbol{x}, \langle \dots, k', k \rangle)\Big]$$

apply recursive definition

$$= \max_{k'} \Big[\alpha_{l,k'} + \boldsymbol{w} \cdot \phi_{l+1}(\boldsymbol{x}, \langle \dots, k', k \rangle)\Big]$$

The Viterbi Algorithm

Runtime $O(LK^2)$

Algorithm 42 $\textsc{ArgmaxForSequences}(\boldsymbol{x}, \boldsymbol{w})$

$L \leftarrow \textsc{len}(\boldsymbol{x})$
$\alpha_{l,k} \leftarrow 0, \quad \zeta_{k,l} \leftarrow 0, \quad \forall\, k = 1 \dots K, \quad \forall l = 0 \dots L$ // initialize variables
for l = $0 \dots L$-1 **do**
  for k = $1 \dots K$ **do**
    $\alpha_{l+1,k} \leftarrow \max_{k'} \left[\alpha_{l,k'} + \boldsymbol{w} \cdot \phi_{l+1}(\boldsymbol{x}, \langle \dots, k', k \rangle)\right]$ // recursion:
    // here, $\phi_{l+1}(\dots k', k \dots)$ is the set of features associated with
    // output position $l+1$ and two adjacent labels k' and k at that position
    $\zeta_{l+1,k} \leftarrow$ the k' that achieves the maximum above // store backpointer
  end for
end for
$\boldsymbol{y} \leftarrow \langle 0, 0, \dots, 0 \rangle$ // initialize predicted output to L-many zeros
$y_L \leftarrow \operatorname{argmax}_k \alpha_{L,k}$ // extract highest scoring final label
for l = L-$1 \dots 1$ **do**
  $y_l \leftarrow \zeta_{l, y_{l+1}}$ // traceback ζ based on y_{l+1}
end for
return $\boldsymbol{y}$ // return predicted output

Key points in Viterbi algorithm

Compute score of best possible prefix up to l+1 ending in k recursively

$$\alpha_{l+1,k} \leftarrow \max_{k'} \left[\alpha_{l,k'} + \boldsymbol{w} \cdot \phi_{l+1}(\boldsymbol{x}, \langle \ldots, k', k \rangle)\right]$$

Record backpointer to label k' in position l that achieves the max

$$\zeta_{l+1,k} = \operatorname*{argmax}_{k'} \left[\alpha_{l,k'} + \boldsymbol{w} \cdot \phi_{l+1}(\boldsymbol{x}, \langle \ldots, k', k \rangle)\right]$$

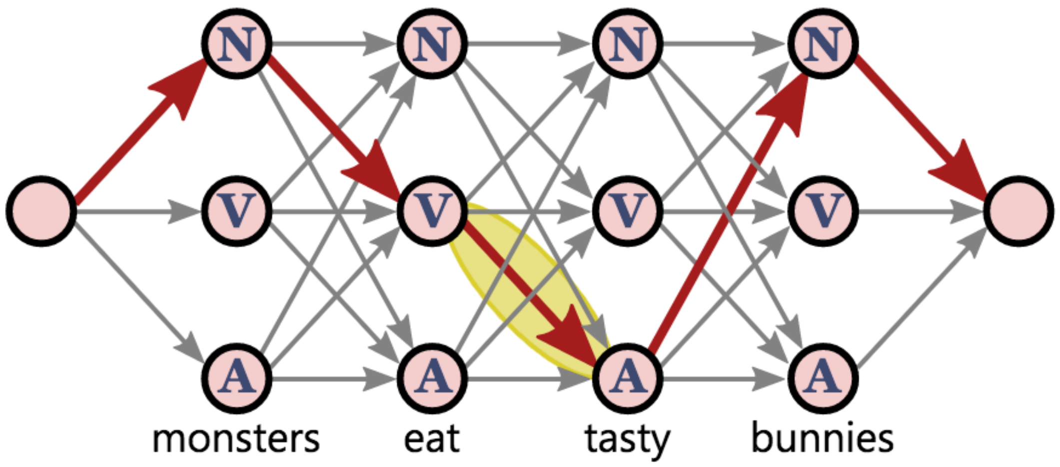
At the end, take $\max_k \alpha_{L,k}$ as the score of the best output sequence

Follow backpointers to retrieve the argmax sequence

Recap: We know how to perform POS tagging with structured perceptron

- An example of sequence labeling tasks
- Requires a predefined set of POS tags
  - Penn Treebank commonly used for English
  - Encodes some distinctions and not others
- Given annotated examples, we can address sequence labeling with multiclass perceptron
  - but computing the argmax naively is expensive
  - constraints on the feature definition make efficient algorithms possible
    - E.g, Viterbi algorithm

Note: one downside of the structured perceptron, we've just seen is that all bad output sequences are equally bad



Consider

$$\widehat{y_1} = [A, A, A, A]$$
$$\widehat{y_2} = [N, V, N, N]$$

- With 0-1 loss
  $l^{(0-1)}(y, \widehat{y_1}) = l^{(0-1)}(y, \widehat{y_2}) = 1$

- An alternative: minimize **Hamming Los**
  - gives a more nuanced evaluation of output than 0–1 loss

$$\ell^{(\text{Ham})}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{l=1}^{L} \mathbf{1}[y_l \neq \hat{y}_l]$$

Can be done with similar algorithms for training and argmax

Sequence labeling tasks

Beyond POS tagging

Many NLP tasks can be framed as sequence labeling

- Information Extraction: detecting named entities
  - E.g., names of people, organizations, locations

"Brendan Iribe, a co-founder of Oculus VR and a prominent University of Maryland donor, is leaving Facebook four years after it purchased his company."

http://www.dbknews.com/2018/10/24/brendan-iribe-facebook-leaves-oculus-vr-umd-computer-science/

Many NLP tasks can be framed as sequence labeling

x = [Brendan, Iribe, “,”, a, co-founder, of, Oculus, VR, and, a, prominent, University, of, Maryland, donor, “,”, is, leaving, Facebook, four, years, after, it, purchased, his, company, “.”]

y = [B-PER, I-PER, O, O, O, O, B-ORG, I-ORG, O, O, O,B-ORG, I-ORG, I-ORG, O, O, O,B-ORG, O, O, O, O, O, O, O, O]

“BIO” labeling scheme for named entity recognition

Many NLP tasks can be framed as sequence labeling

- The same kind of BIO scheme can be used to tag other spans of text
  - Syntactic analysis: detecting noun phrase and verb phrases
  - Semantic roles: detecting semantic roles (who did what to whom)