COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

Sequence Labeling with the Structured Perceptron

CMSC 470

Marine Carpuat

POS tagging
Sequence labeling with the perceptron

Sequence labeling problem

- Input:
  - sequence of tokens $x = [x_1 \dots x_L]$
  - Variable length L
- Output (aka label):
  - sequence of tags $y = [y_1 \dots y_L]$
  - # tags = K
  - Size of output space?

Structured Perceptron

- Perceptron algorithm can be used for sequence labeling
- But there are challenges
  - How to compute argmax efficiently?
  - What are appropriate features?
- Approach: leverage structure of output space

Perceptron algorithm remains the same as for multiclass classification

$$\hat{y} = \arg\max_{y} \boldsymbol{\theta}^{\mathsf{T}} \mathbf{f}(\mathbf{x}, y)$$

Algorithm 3 Perceptron learning algorithm

procedure PERCEPTRON($\boldsymbol{x}^{(1:N)}, y^{(1:N)}$)
  $t \leftarrow 0$
  $\boldsymbol{\theta}^{(0)} \leftarrow \mathbf{0}$
  repeat
    $t \leftarrow t + 1$
    Select an instance i
    $\hat{y} \leftarrow \text{argmax}_y\, \boldsymbol{\theta}^{(t-1)} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, y)$
    if $\hat{y} \neq y^{(i)}$ **then**
      $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} + \boldsymbol{f}(\boldsymbol{x}^{(i)}, y^{(i)}) - \boldsymbol{f}(\boldsymbol{x}^{(i)}, \hat{y})$
    else
      $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)}$
  until tired
  return $\boldsymbol{\theta}^{(t)}$

Note: CIML denotes
- the weight vector as w instead of θ
- The feature function as $\Phi(x, y)$ instead of $f(x, y)$

Feature functions for sequence labeling

$$\boldsymbol{x} = \text{“ monsters eat tasty bunnies “}$$
$$\boldsymbol{y} = \quad \text{noun} \;\; \text{verb} \;\; \text{adj} \;\; \text{noun}$$

- Standard features of POS tagging
  - **Unary features:** # times word w has been labeled with tag l for all words w and all tags l
  - **Markov features:** # times tag l is adjacent to tag l' in output for all tags l and l'
- Size of feature representation is constant wrt input length

Example from CIML chapter 17

Solving the argmax problem for sequences with dynamic programming

$\boldsymbol{x} =$ " monsters eat tasty bunnies "

$\boldsymbol{y} =$ noun verb adj noun

- Efficient algorithms possible if **the feature function decomposes over the input**
- This holds for unary and markov features used for POS tagging

Decomposition of structure

- Features decompose over the input if $\phi(\boldsymbol{x}, \boldsymbol{y}) = \sum_{l=1}^{L} \phi_l(\boldsymbol{x}, \boldsymbol{y})$

Feature function that only includes features about position l

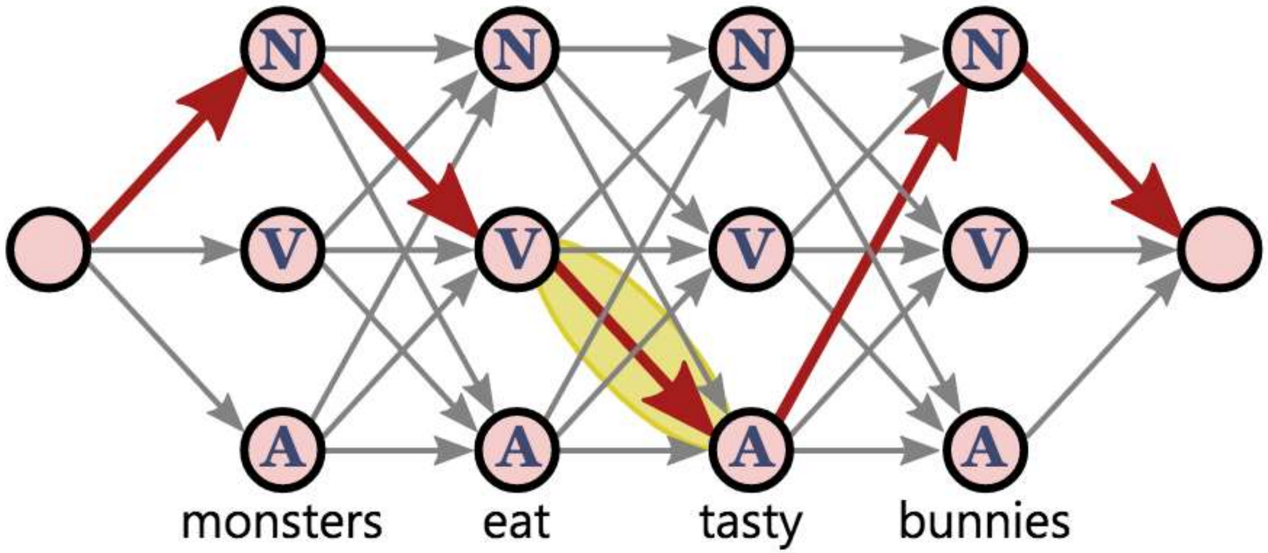
- If features decompose over the input, structures (x,y) can be scored incrementally

$$\begin{aligned} \boldsymbol{w} \cdot \phi(\boldsymbol{x}, \boldsymbol{y}) &= \boldsymbol{w} \cdot \sum_{l=1}^{L} \phi_l(\boldsymbol{x}, \boldsymbol{y}) && \text{decomposition of structure} \\ &= \sum_{l=1}^{L} \boldsymbol{w} \cdot \phi_l(\boldsymbol{x}, \boldsymbol{y}) && \text{associative law} \end{aligned}$$

Decomposition of structure: Lattice/trellis representation

$x =$ " monsters eat tasty bunnies "

$y =$ noun verb adj noun



- Trellis sequence labeling
  - Any path represents a labeling of input sentence
  - Gold standard path in red
  - Each edge receives a weight such that adding weights along the path corresponds to score for input/ouput configuration
- Any max-weight path algorithm can find the argmax
  - We'll describe the Viterbi algorithm

Dynamic programming solution relies on recursively computing prefix scores $\alpha_{l,k}$

Score of best possible output prefix, up to and including position l, that labels the l-th word as label k

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

Sequence of labels of length l-1

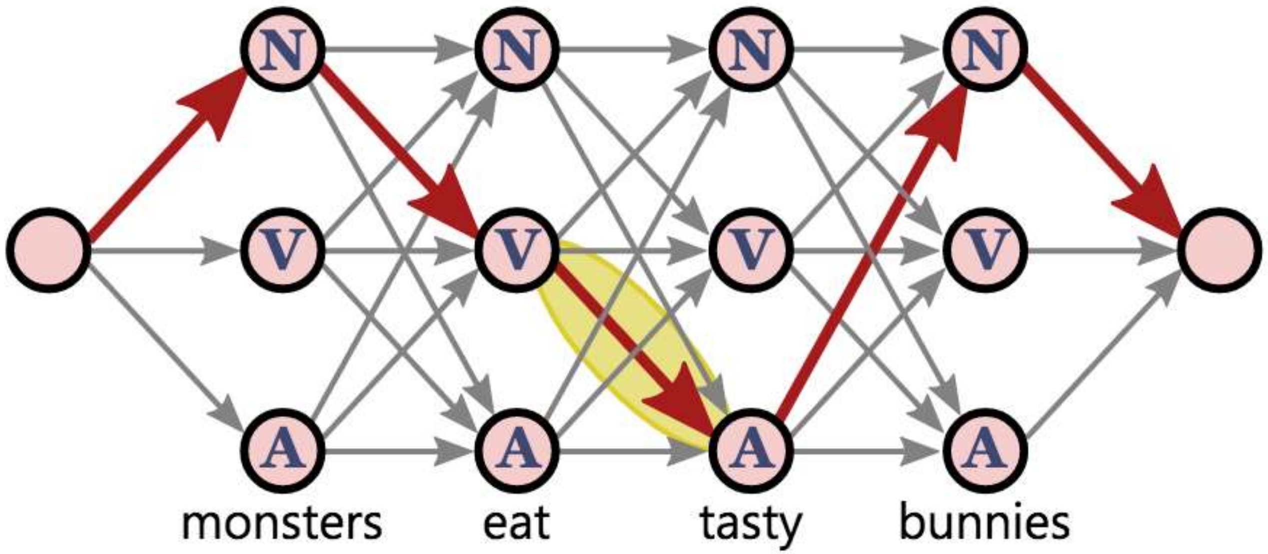
Features for sequence starting at position 1 up to and including position l

Sequence of length l obtained by adding k at the end.

Computing prefix scores $\alpha_{l,k}$ Example

$\boldsymbol{x} =$ " monsters eat tasty bunnies "

$\boldsymbol{y} =$ noun verb adj noun



Let's compute $\alpha_{3,A}$ given

- Prefix scores for length 2

$$\alpha_{2,N} = 2, \alpha_{2,V} = 9, \alpha_{2,A} = -1$$

- Unary feature weights

$$w_{tasty/A} = 1.2$$

- Markov feature weights

$$w_{N,A} = -5,\ w_{V,A} = 2.5, w_{A,A} = \ 2.2$$

Dynamic programming solution relies on recursively computing prefix scores $\alpha_{l,k}$

Score of best possible output prefix, up to and including position l+1, that labels the (l+1)-th word as label k

Backpointer to the label that achieves the above maximum

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

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

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

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

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

Derivation on board + CIML ch17

Viterbi algorithm

Assumptions:

- Unary features
- Markov features based on 2 adjacent labels

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 $l = 0 \dots L\text{-}1$ **do**
  for $k = 1 \dots K$ **do**
    $\alpha_{l+1,k} \leftarrow \max_{k'} \left[\alpha_{l,k'} + \boldsymbol{w} \cdot \boldsymbol{\phi}_{l+1}(\boldsymbol{x}, \langle \dots, k', k \rangle) \right]$ // 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 $l = L\text{-}1 \dots 1$ **do**
  $y_l \leftarrow \zeta_{l, y_{l+1}}$ // traceback ζ based on y_{l+1}
end for
return $\boldsymbol{y}$ // return predicted output

Exercise: Impact of feature definitions

- Consider a structured perceptron with the following features
  - # times word w has been labeled with tag l for all words w and all tags l
  - # times word w has been labeled with tag l when it follows word w’ for all words w, w’ and all tags l
  - # times tag l occurs in the sequence (l’,l’’,l) in the output for all tags l, l’, l’’
- What is the dimension of the perceptron weight vector?
- Can we use dynamic programming to compute the argmax?

Recap: POS tagging

- 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
  - Viterbi algorithm for unary and markov features