Sequence Labeling with the Structured Perceptron

CMSC 470
Marine Carpuat
Sequence labeling problem

• Input:
  • sequence of tokens $x = [x_1 \ldots x_L]$  
  • Variable length $L$

• Output (aka label):
  • sequence of tags $y = [y_1 \ldots y_L]$  
  • # 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

\[ \hat{y} = \arg \max_y \theta^T f(x, y) \]

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

<table>
<thead>
<tr>
<th>Algorithm 3 Perceptron learning algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: procedure PERCEPTRON(( x^{(1:N)} ), ( y^{(1:N)} ))</td>
</tr>
<tr>
<td>2: ( t \leftarrow 0 )</td>
</tr>
<tr>
<td>3: ( \theta^{(0)} \leftarrow 0 )</td>
</tr>
<tr>
<td>4: repeat</td>
</tr>
<tr>
<td>5: ( t \leftarrow t + 1 )</td>
</tr>
<tr>
<td>6: Select an instance ( i )</td>
</tr>
<tr>
<td>7: ( \hat{y} \leftarrow \arg \max_y \theta^{(t-1)} \cdot f(x^{(i)}, y) )</td>
</tr>
<tr>
<td>8: if ( \hat{y} \neq y^{(i)} ) then</td>
</tr>
<tr>
<td>9: ( \theta^{(t)} \leftarrow \theta^{(t-1)} + f(x^{(i)}, y^{(i)}) - f(x^{(i)}, \hat{y}) )</td>
</tr>
<tr>
<td>10: else</td>
</tr>
<tr>
<td>11: ( \theta^{(t)} \leftarrow \theta^{(t-1)} )</td>
</tr>
<tr>
<td>12: until tired</td>
</tr>
<tr>
<td>13: return ( \theta^{(t)} )</td>
</tr>
</tbody>
</table>
Feature functions for sequence labeling

- 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

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

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

Example from CIML chapter 17
Solving the argmax problem for sequences with dynamic programming

$x = " \text{monsters eat tasty bunnies}"$
$y = \text{noun, verb, adj, noun}$

- 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
  \[ \phi(x, y) = \sum_{l=1}^{L} \phi_l(x, y) \]

• If features decompose over the input, structures \((x,y)\) can be scored incrementally
  \[ w \cdot \phi(x, y) = w \cdot \sum_{l=1}^{L} \phi_l(x, y) \]
  
  \[ = \sum_{l=1}^{L} w \cdot \phi_l(x, y) \]

  decomposition of structure

  associative law
Decomposition of structure: Lattice/trellis representation

- 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/output configuration

- Any max-weight path algorithm can find the argmax
  - We’ll describe the Viterbi algorithm

\[ x = \text{"monsters eat tasty bunnies"} \]
\[ y = \text{noun verb adj noun} \]
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 $k$

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

- 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**

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 $\alpha_{l,k}$

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

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

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

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

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

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
Viterbi algorithm

Assumptions:
- Unary features
- Markov features based on 2 adjacent labels

Runtime: $O(LK^2)$

**Algorithm 42 ArgmaxForSequences($x, w$)**

1. $L \leftarrow \text{len}(x)$
2. $\alpha_{l,k} \leftarrow 0, \quad \zeta_{l,k} \leftarrow 0, \quad \forall k = 1 \ldots K, \quad \forall l = 0 \ldots L$ // initialize variables
3. for $l = 0 \ldots L-1$ do
4.   for $k = 1 \ldots K$ do
5.     $\alpha_{l+1,k} \leftarrow \max_{k'} \left[ \alpha_{l,k'} + w \cdot \phi_{l+1}(x, (\ldots, k', k)) \right]$ // recursion:
6.       // here, $\phi_{l+1}(\ldots, k', k, \ldots)$ is the set of features associated with
7.       // output position $l + 1$ and two adjacent labels $k'$ and $k$ at that position
8.     $\zeta_{l+1,k} \leftarrow$ the $k'$ that achieves the maximum above // store backpointer
9.   end for
10. end for
11. $y \leftarrow \langle 0, 0, \ldots, 0 \rangle$ // initialize predicted output to $L$-many zeros
12. $y_L \leftarrow \text{argmax}_k \alpha_{L,k}$ // extract highest scoring final label
13. for $l = L-1 \ldots 1$ do
14.   $y_{l} \leftarrow \zeta_{l,y_{l+1}}$ // traceback $\zeta$ based on $y_{l+1}$
15. end for
16. return $y$ // 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