Sequence Labeling: more tasks, more methods

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
  - Viterbi algorithm for unary and markov features
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.”

Many NLP tasks can be framed as sequence labeling

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


“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)
Many NLP tasks can be framed as sequence labeling

• Other sequence labeling tasks
  • Language identification in code-switched text
    “Ulikuwa ukiongea a lot of nonsense.” (Swahili/English)
  • Metaphor detection
    “he swam in a sea of diamonds”
    “authority is a chair, it needs legs to stand”
    “in Washington, people change dance partners frequently, but not the dance”
• …
Other algorithms for solving the argmax problem
Structured perceptron can be used for other structures than sequences

- The Viterbi algorithm we’ve seen is specific to sequences
  - Other argmax algorithms necessary for other structures (e.g. trees)

- Integer Linear Programming provides a general framework for solving the argmax problem
Argmax problem as an Integer Linear Program

- An integer linear program (ILP) is an optimization problem of the form

\[
\max_{z} \quad a \cdot z \quad \text{subj. to} \quad \text{linear constraints on } z
\]

- For a fixed vector \(a\)
  - Example of integer constraint: \(z_3 \in \{0, 1\}\)

- Well-engineered solvers exist
  - e.g, Gurobi
  - Useful for prototyping
  - But general not as efficient as dynamic programming
Casting sequence labeling with Markov features as an ILP

• Step 1: Define variables $z$ as binary indicator variables which encode an output sequence $y$

$$z_{l,k',k} = 1[\text{label } l \text{ is } k \text{ and label } l - 1 \text{ is } k']$$

• Step 2: Construct the linear objective function

$$a_{l,k',k} = \mathbf{w} \cdot \phi_l(x, \langle \ldots, k', k \rangle)$$
Casting sequence labeling with Markov features as an ILP

• Step 3: Define constraints to ensure a well-formed solution
  • Z’s should be binary: for all $l, k', k$
    $$z_{l,k',k} \in \{0, 1\}$$
  • For a given position $l$, there is exactly one active $z$
    $$\sum_k \sum_{k'} z_{l,k',k} = 1 \text{ for all } l$$
  • The $z$’s are internally consistent
    $$\sum_{k'} z_{l,k',k} = \sum_{k''} z_{l+1,k,k''} \text{ for all } l, k$$
Loss-augmented structured prediction
In default structured perceptron, all bad output sequences are equally bad.

With 0-1 loss:
\[ l^{(0-1)}(y, \hat{y}_1) = l^{(0-1)}(y, \hat{y}_2) = 1 \]

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

\[ \ell^{(\text{Ham})}(y, \hat{y}) = \sum_{l=1}^{L} 1[y_l \neq \hat{y}_l] \]
Loss functions for structured prediction

• Recall learning as optimization for multiclass classification
  
  • e.g., \( \min_w \frac{1}{2} \|w\|^2 + C \sum_n \ell^{(\text{hin})}(y_n, w \cdot x_n + b) \)

• Let’s define a structure-aware optimization objective

  • e.g., \( \min_w \frac{1}{2} \|w\|^2 + C \sum_n \ell^{(s-h)}(y_n, x_n, w) \)

  \[ \ell^{(s-h)}(y_n, x_n, w) = \max \left\{ 0, \max_{\hat{y} \in \mathcal{Y}(x_n)} \left[ s_w(x_n, \hat{y}) + \ell^{(\text{Ham})}(y_n, \hat{y}) \right] - s_w(x_n, y_n) \right\} \]

Structured hinge loss

• 0 if true output beats score of every imposter output
• Otherwise: scales linearly as function of score diff between most confusing imposter and true output
Optimization: stochastic subgradient descent

- Subgradients of structured hinge loss?

\[
\nabla_w \ell^{(s-h)}(y, x, w) \quad \text{if the loss is } > 0
\]

\begin{align}
\nabla_w \ell^{(s-h)}(y, x, w) &= \max_{\hat{y} \in \mathcal{Y}(x_n)} \left[ w \cdot \phi(x_n, \hat{y}) + \ell(y_n, \hat{y}) \right] - w \cdot \phi(x_n, y_n) \\
&= \nabla_w \max_{\hat{y} \in \mathcal{Y}(x_n)} \left[ w \cdot \phi(x_n, \hat{y}) + \ell(y_n, \hat{y}) \right] - w \cdot \phi(x_n, y_n) \\
&= \nabla_w \left[ w \cdot \phi(x_n, \hat{y}) - w \cdot \phi(x_n, y_n) + \ell(y_n, \hat{y}) \right] \\
&= \phi(x_n, \hat{y}) - \phi(x_n, y_n)
\end{align}

(17.25) (17.26) (17.27) (17.28)
Optimization: stochastic subgradient descent

• subgradients of structured hinge loss

\[ \nabla_w \ell^{(s-h)}(y_n, x_n, w) = \begin{cases} 
0 & \text{if } \ell^{(s-h)}(y_n, x_n, w) = 0 \\
\phi(x_n, \hat{y}_n) - \phi(x_n, y_n) & \text{otherwise}
\end{cases} \]

where \( \hat{y}_n = \arg\max_{\hat{y}_n \in \mathcal{Y}(x_n)} [w \cdot \phi(x_n, \hat{y}_n) + \ell(y_n, \hat{y}_n)] \)  \( (17.29) \)
Optimization: stochastic subgradient descent
Resulting training algorithm

---

**Algorithm 41** _StochSubGradStructSVM(\(D, MaxIter, \lambda, \ell\))_

1. \(w \leftarrow 0\) // initialize weights
2. for iter = 1 \(\ldots\) MaxIter do
3.     for all \((x, y) \in D\) do
4.         \(\hat{y} \leftarrow \arg\max_{\hat{y} \in \mathcal{Y}(x)} w \cdot \phi(x, \hat{y}) + \ell(y, \hat{y})\) // loss-augmented prediction
5.         if \(\hat{y} \neq y\) then
6.             \(w \leftarrow w + \phi(x, y) - \phi(x, \hat{y})\) // update weights
7.         end if
8.     \(w \leftarrow w - \frac{\lambda}{N}w\) // shrink weights due to regularizer
9. end for
10. end for
11. return \(w\) // return learned weights

---

Only 2 differences compared to structured perceptron!
Loss-augmented inference/search
Recall dynamic programming solution without Hamming loss

\[ \tilde{\alpha}_{l+1,k} = \max_{\hat{y}_{1:l}} \omega \cdot \phi_{1:l+1}(x, \hat{y} \circ k) \]

\[ = \max_{k'} \left[ \tilde{\alpha}_{l,k'} + \omega \cdot \phi_{l+1}(x, \{\ldots, k', k\}) \right] \]
Loss-augmented inference/search
Dynamic programming with Hamming loss

\[ \tilde{\alpha}_{l+1,k} = \max_{\hat{y}_{1:l}} \omega \cdot \phi_{1:l+1}(x, \hat{y} \circ k) + \ell_{1:l+1}^{(Ham)}(y, \hat{y} \circ k) \]

\[ = \max_{k'} \left[ \tilde{\alpha}_{l,k'} + \omega \cdot \phi_{l+1}(x, \langle \ldots, k', k \rangle) \right] + 1[k \neq y_{l+1}] \]

We can use Viterbi algorithm as before as long as the loss function decomposes over the input consistently with features!
Sequence labeling

• Structured perceptron
  • A general algorithm for structured prediction problems such as sequence labeling

• The Argmax problem
  • Efficient argmax for sequences with Viterbi algorithm, given some assumptions on feature structure
  • A more general solution: Integer Linear Programming

• Loss-augmented structured prediction
  • Training algorithm
  • Loss-augmented argmax