Natural Language Processing (CSE 447/547M): Sequence Models

Noah Smith

University of Washington
nasmith@cs.washington.edu

February 6, 2019

Where We Are

- Language models
- Text classification
- Next: **linguistic analysis**

Linguistic Analysis: Overview

Every linguistic analyzer is comprised of:

1. Theoretical motivation from linguistics and/or the text domain
2. An algorithm that maps $\mathcal{V}^\dagger$ to some output space $\mathcal{Y}$.
   - In this class, I'll start with abstract algorithms applicable to many problems.
3. An implementation of the algorithm
   - Once upon a time: rule systems and crafted rules
   - Most common now: supervised learning from annotated data
   - Frontier: less supervision (semi-, un-, distant, ...)

Sequence Labeling

After text classification ($\mathcal{V}^\dagger \to \mathcal{L}$), the next simplest type of output is a **sequence labeling**.

$$\langle x_1, x_2, \ldots, x_\ell \rangle \mapsto \langle y_1, y_2, \ldots, y_\ell \rangle$$

Every word (or character) gets a label in $\mathcal{L}$.
Example problems:

- part-of-speech tagging (Church, 1988)
- spelling correction (Kernighan et al., 1990)
- word alignment (Vogel et al., 1996)
- named-entity recognition (Bikel et al., 1999)
- compression (Conroy and O'Leary, 2001)

Version 0: The Simplest Sequence Labeler

Define score of a labeled word in context: $s(\boldsymbol{x}, i, y)$, for example through a feature vector, $\boldsymbol{\phi}(\boldsymbol{x}, i, y)$.

Train a classifier, e.g.,

$$\hat{y}_i = \operatorname*{argmax}_{y \in \mathcal{L}} s(\boldsymbol{x}, i, y)$$

$$\stackrel{\text{linear}}{=} \operatorname*{argmax}_{y \in \mathcal{L}} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, i, y)$$

Version 0: The Simplest Sequence Labeler

Define score of a labeled word in context: $s(\boldsymbol{x}, i, y)$, for example through a feature vector, $\boldsymbol{\phi}(\boldsymbol{x}, i, y)$.

Train a classifier, e.g.,

$$\begin{aligned}\hat{y}_i &= \operatorname*{argmax}_{y \in \mathcal{L}} s(\boldsymbol{x}, i, y) \\ &\overset{\text{linear}}{=} \operatorname*{argmax}_{y \in \mathcal{L}} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, i, y)\end{aligned}$$

Sometimes this works! E.g., one or two-layer neural network on top of ELMo contextual word vectors (features of $\boldsymbol{x}$).

Version 0: The Simplest Sequence Labeler

Define score of a labeled word in context: $s(\boldsymbol{x}, i, y)$, for example through a feature vector, $\boldsymbol{\phi}(\boldsymbol{x}, i, y)$.

Train a classifier, e.g.,

$$\begin{aligned}\hat{y}_i &= \operatorname*{argmax}_{y \in \mathcal{L}} s(\boldsymbol{x}, i, y) \\ &\overset{\text{linear}}{=} \operatorname*{argmax}_{y \in \mathcal{L}} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, i, y)\end{aligned}$$

Sometimes this works! E.g., one or two-layer neural network on top of ELMo contextual word vectors (features of $\boldsymbol{x}$).

We can do better when there are predictable relationships between Y_i and Y_{i+1}.

Generative Sequence Labeling (Version 1): Hidden Markov Models

Note: small update relative to lecture; "Y_0" removed.

$$p(\boldsymbol{x}, \boldsymbol{y}) = \pi_{y_1} \prod_{i=1}^{\ell} \theta_{x_i | y_i} \cdot \gamma_{y_{i+1} | y_i}$$

For each state/label $y \in \mathcal{L}$:

- π_y is the start probability; $p(Y_1 = y)$
- $\boldsymbol{\theta}_{*|y}$ is the "emission" distribution; $\theta_{x_i|y_i} = p(X_i = x_i \mid Y_i = y_i)$
- $\boldsymbol{\gamma}_{*|y}$ is called the "transition" distribution; $\gamma_{y_{i+1}|y_i} = p(Y_{i+1} = y_{i+1} \mid Y_i = y_i)$
- By convention, $y_{\ell+1} = \bigcirc$ is always the "stop state"

A More General Form

Twice now, we've made the move from generative models based on repeated "rolls of dice" to discriminative models based on feature representations.

- Language modeling
- Text classification (naïve Bayes)

In the HMM case, we would like to do the same thing:

$$\begin{aligned}\hat{\boldsymbol{y}} &= \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^\ell} p(y_1) \prod_{i=1}^{\ell} p(x_i \mid y_i) \cdot p(y_{i+1} \mid y_i) \\ &= \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^\ell} \underbrace{\log p(y_1)}_{s(\bigcirc, y_1)} + \sum_{i=1}^{\ell} \underbrace{\log p(x_i \mid y_i)}_{s(x_i, y_i)} + \underbrace{\log p(y_{i+1} \mid y_i)}_{s(y_i, y_{i+1})}\end{aligned}$$

In this case, each Y_i "interacts" with Y_{i-1} and Y_{i+1} directly.

HMMs incorporate "structure."

"Simplest sequence labeler" (version 0):

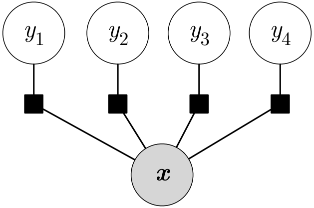
$$\forall i \in \{1, \ldots, \ell\}, \quad \hat{y}_i = \operatorname*{argmax}_{y \in \mathcal{L}} s(\boldsymbol{x}, i, y)$$

HMM-style sequence labeler (version 1):

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^\ell} s(\bigcirc, y_1) + \sum_{i=1}^{\ell} s(x_i, y_i) + s(y_i, y_{i+1})$$

HMMs incorporate "structure."

"Simplest sequence labeler" (version 0):



$$\forall i \in \{1, \ldots, \ell\}, \quad \hat{y}_i = \operatorname*{argmax}_{y \in \mathcal{L}} s(\boldsymbol{x}, i, y)$$

HMM-style sequence labeler (version 1):

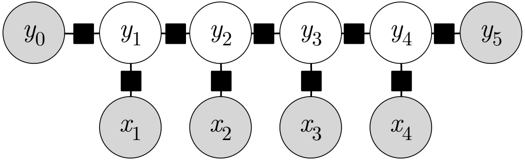
$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^\ell} s(\bigcirc, y_1) + \sum_{i=1}^{\ell} s(x_i, y_i) + s(y_i, y_{i+1})$$

HMMs incorporate "structure."

"Simplest sequence labeler" (version 0):

$$\forall i \in \{1, \ldots, \ell\}, \quad \hat{y}_i = \operatorname*{argmax}_{y \in \mathcal{L}} s(\boldsymbol{x}, i, y)$$

HMM-style sequence labeler (version 1):



$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^\ell} s(\bigcirc, y_1) + \sum_{i=1}^{\ell} s(x_i, y_i) + s(y_i, y_{i+1})$$

HMMs incorporate "structure."

"Simplest sequence labeler" (version 0):

$$\forall i \in \{1, \ldots, \ell\}, \quad \hat{y}_i = \operatorname*{argmax}_{y \in \mathcal{L}} s(\boldsymbol{x}, i, y)$$

HMM-style sequence labeler (version 1):

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^\ell} s(\bigcirc, y_1) + \sum_{i=1}^{\ell} s(x_i, y_i) + s(y_i, y_{i+1})$$

Each of these has an advantage over the other:

- The simple unstructured classifier makes all decisions independently but can "see" all the inputs.
- The structured version lets the different labels "interact."

A More Powerful Solution (Version 2)

Slightly more generally, define scores of adjacent labels in context: $s(\boldsymbol{x}, i, y_i, y_{i+1})$.

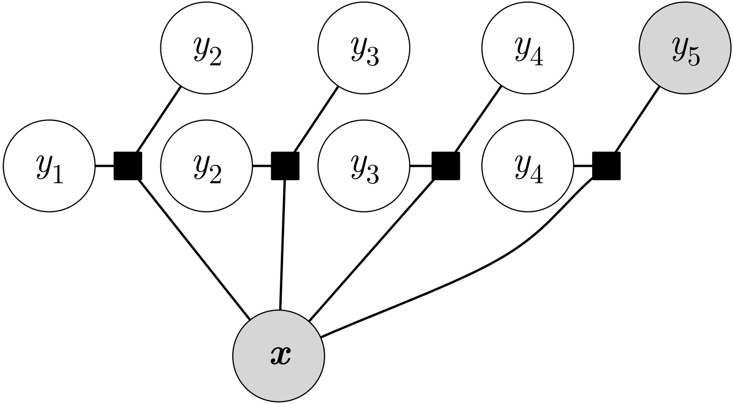
Features can depend on *any words at all*; this turns out not to affect asymptotic cost of prediction!

Local Pairwise Classifier

$$(\hat{y}_i, \hat{y}_{i+1}) = \operatorname*{argmax}_{y', y \in \mathcal{L}} s(\boldsymbol{x}, i, y', y)$$

Local Pairwise Classifier

$$(\hat{y}_i, \hat{y}_{i+1}) = \operatorname*{argmax}_{y', y \in \mathcal{L}} s(\boldsymbol{x}, i, y', y)$$



Local Pairwise Classifier

$$(\hat{y}_i, \hat{y}_{i+1}) = \operatorname*{argmax}_{y', y \in \mathcal{L}} s(\boldsymbol{x}, i, y', y)$$

The problem is with disagreements: what if the $Y_{1:2}$ prediction and the $Y_{2:3}$ prediction do not agree about Y_2?

Even More Powerful: "Global" Prediction (Version 2)

Still version 2, defining scores of adjacent word-labels in context: $s(\boldsymbol{x}, i, y', y)$

But now we have a better, "structured" classifer/predictor:

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^{\ell}} \sum_{i=0}^{\ell} s(\boldsymbol{x}, i, y_i, y_{i+1})$$

(By convention, $y_0 = \bigcirc$ is the fixed "start state." $y_{\ell+1} =$ is still the stop state.)

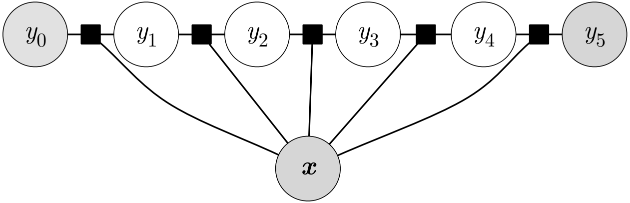
Even More Powerful: "Global" Prediction (Version 2)

Still version 2, defining scores of adjacent word-labels in context: $s(\boldsymbol{x}, i, y', y)$

But now we have a better, "structured" classifer/predictor:

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^\ell} \sum_{i=0}^{\ell} s(\boldsymbol{x}, i, y_i, y_{i+1})$$

(By convention, $y_0 = \bigcirc$ is the fixed "start state." $y_{\ell+1} = \octagon$ is still the stop state.)



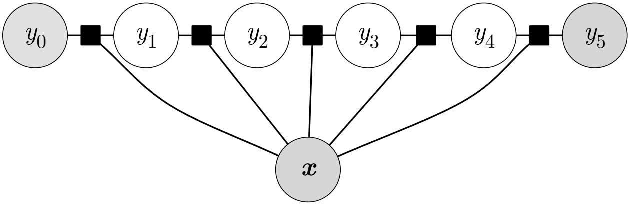
Even More Powerful: "Global" Prediction (Version 2)

Still version 2, defining scores of adjacent word-labels in context: $s(\boldsymbol{x}, i, y', y)$

But now we have a better, "structured" classifer/predictor:

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^\ell} \sum_{i=0}^{\ell} s(\boldsymbol{x}, i, y_i, y_{i+1})$$

(By convention, $y_0 = \bigcirc$ is the fixed "start state." $y_{\ell+1} =$ ⯃ is still the stop state.)



This is a fundamentally different kind of problem, demanding new:

- predicting ("decoding") algorithms
- training algorithms (to be discussed later)

Prediction with HMMs

~~We'll start with the classical HMM (version 1), then return later to the version 2 case.~~
You saw the classical HMM in section. Lecture will show the version 2 case.

$$\operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^\ell} p(y_1) \prod_{i=1}^{\ell} p(x_i \mid y_i) \cdot p(y_{i+1} \mid y_i)$$

How to optimize over $|\mathcal{L}|^\ell$ choices without explicit enumeration?

Prediction with HMMs

~~We'll start with the classical HMM (version 1), then return later to the version 2 case.~~

You saw the classical HMM in section. Lecture will show the version 2 case.

$$\operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{L}^\ell} p(y_1) \prod_{i=1}^{\ell} p(x_i \mid y_i) \cdot p(y_{i+1} \mid y_i)$$

How to optimize over $|\mathcal{L}|^\ell$ choices without explicit enumeration?

Key: exploit the conditional independence assumptions:

$$Y_i \perp \boldsymbol{Y}_{1:i-2} \mid Y_{i-1}$$
$$Y_i \perp \boldsymbol{Y}_{i+2:\ell} \mid Y_{i+1}$$

Part-of-Speech Tagging Example

	I	suspect	the	present	forecast	is	pessimistic	.
noun	•	•	•	•	•	•		
adj.		•		•	•		•	
adv.				•				
verb		•		•	•	•		
num.	•							
det.			•					
punc.								•

With this very simple tag set, $7^8 = 5.7$ million labelings.
(Even restricting to the possibilities above, 288 labelings.)

References I

Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. An algorithm that learns what's in a name. *Machine learning*, 34(1–3):211–231, 1999.

Kenneth W. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proc. of ANLP*, 1988.

John M. Conroy and Dianne P. O'Leary. Text summarization via hidden Markov models. In *Proc. of SIGIR*, 2001.

Mark D. Kernighan, Kenneth W. Church, and William A. Gale. A spelling correction program based on a noisy channel model. In *Proc. of COLING*, 1990.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. HMM-based word alignment in statistical translation. In *Proc. of COLING*, 1996.