Global Linear Models Part 2:
The Perceptron Algorithm for Tagging

Michael Collins, Columbia University

Recap: Three Components of Global Linear Models

- $\mathbf{f}$ is a function that maps a structure (x, y) to a **feature vector** $\mathbf{f}(x, y) \in \mathbb{R}^d$
- $\mathbf{GEN}$ is a function that maps an input x to a set of **candidates** $\mathbf{GEN}(x)$
- $\mathbf{v}$ is a parameter vector (also a member of $\mathbb{R}^d$)
- Training data is used to set the value of $\mathbf{v}$

Recap: Putting it all Together

- $\mathcal{X}$ is set of sentences, $\mathcal{Y}$ is set of possible outputs (e.g. trees)
- Need to learn a function $F : \mathcal{X} \rightarrow \mathcal{Y}$
- $\mathbf{GEN}$, $\mathbf{f}$, $\mathbf{v}$ define

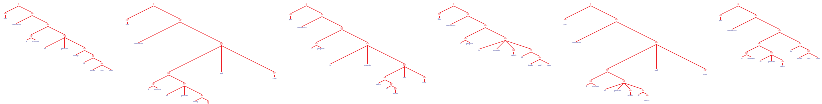
$$F(x) = \arg\max_{y \in \mathbf{GEN}(x)} \mathbf{f}(x, y) \cdot \mathbf{v}$$

Choose the highest scoring candidate as the most plausible structure

- Given examples (x_i, y_i), how to set $\mathbf{v}$?

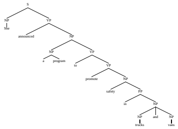
She announced a program to promote safety in trucks and vans

$\Downarrow$ **GEN**



$\Downarrow \mathbf{f}$	$\Downarrow \mathbf{f}$	$\Downarrow \mathbf{f}$	$\Downarrow \mathbf{f}$	$\Downarrow \mathbf{f}$	$\Downarrow \mathbf{f}$
$\langle 1, 1, 3, 5 \rangle$	$\langle 2, 0, 0, 5 \rangle$	$\langle 1, 0, 1, 5 \rangle$	$\langle 0, 0, 3, 0 \rangle$	$\langle 0, 1, 0, 5 \rangle$	$\langle 0, 0, 1, 5 \rangle$
$\Downarrow \mathbf{f} \cdot \mathbf{v}$	$\Downarrow \mathbf{f} \cdot \mathbf{v}$	$\Downarrow \mathbf{f} \cdot \mathbf{v}$	$\Downarrow \mathbf{f} \cdot \mathbf{v}$	$\Downarrow \mathbf{f} \cdot \mathbf{v}$	$\Downarrow \mathbf{f} \cdot \mathbf{v}$
13.6	12.2	12.1	3.3	9.4	11.1

$\Downarrow \arg\max$



Recap: A Variant of the Perceptron Algorithm

Inputs: Training set (x_i, y_i) for $i = 1 \ldots n$

Initialization: $\mathbf{v} = 0$

Define: $F(x) = \mathrm{argmax}_{y \in \mathbf{GEN}(x)} \mathbf{f}(x, y) \cdot \mathbf{v}$

Algorithm: For $t = 1 \ldots T$, $i = 1 \ldots n$

$z_i = F(x_i)$

If $(z_i \neq y_i)$ $\quad \mathbf{v} = \mathbf{v} + \mathbf{f}(x_i, y_i) - \mathbf{f}(x_i, z_i)$

Output: Parameters $\mathbf{v}$

Tagging Problems

TAGGING: Strings to Tagged Sequences

a b e e a f h j ⇒ a/C b/D e/C e/C a/D f/C h/D j/C

Example 1: Part-of-speech tagging

Profits/N soared/V at/P Boeing/N Co./N ,/, easily/ADV topping/V forecasts/N on/P Wall/N Street/N ,/, as/P their/POSS CEO/N Alan/N Mulally/N announced/V first/ADJ quarter/N results/N ./.

Example 2: Named Entity Recognition

Profits/NA soared/NA at/NA Boeing/SC Co./CC ,/NA easily/NA topping/NA forecasts/NA on/NA Wall/SL Street/CL ,/NA as/NA their/NA CEO/NA Alan/SP Mulally/CP announced/NA first/NA quarter/NA results/NA ./NA

Tagging using Global Linear Models

- Inputs x are sentences $w_{[1:n]} = \{w_1 \dots w_n\}$
- Define $\mathcal{T}$ to be the set of possible tags
- $\mathbf{GEN}(w_{[1:n]}) = \mathcal{T}^n$ i.e. all tag sequences of length n
- Note: The size of $\mathbf{GEN}$ is exponential in the sentence length
- How do we define $\mathbf{f}$?

Representation: Histories

- A **history** is a 4-tuple $\langle t_{-2}, t_{-1}, w_{[1:n]}, i \rangle$
- t_{-2}, t_{-1} are the previous two tags.
- $w_{[1:n]}$ are the n words in the input sentence.
- i is the index of the word being tagged

Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/?? from which Spain expanded its empire into the rest of the Western Hemisphere .

- $t_{-2}, t_{-1} =$ DT, JJ
- $w_{[1:n]} = \langle Hispaniola, quickly, became, \ldots, Hemisphere, . \rangle$
- $i = 6$

Local Feature-Vector Representations

- Take a history/tag pair (h, t).
- $g_s(h, t)$ for $s = 1 \ldots d$ are **local features** representing tagging decision t in context h.

Example: POS Tagging

- **Word/tag features**

$$g_{100}(h, t) = \begin{cases} 1 & \text{if current word } w_i \text{ is } \texttt{base} \text{ and } t = \texttt{VB} \\ 0 & \text{otherwise} \end{cases}$$

$$g_{101}(h, t) = \begin{cases} 1 & \text{if current word } w_i \text{ ends in } \texttt{ing} \text{ and } t = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

- **Contextual Features**

$$g_{103}(h, t) = \begin{cases} 1 & \text{if } \langle t_{-2}, t_{-1}, t \rangle = \langle \text{DT, JJ, VB} \rangle \\ 0 & \text{otherwise} \end{cases}$$

A tagged sentence with n words has n history/tag pairs

Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/NN

History				Tag
t_{-2}	t_{-1}	$w_{[1:n]}$	i	t
*	*	$\langle Hispaniola, quickly, \ldots, \rangle$	1	NNP
*	NNP	$\langle Hispaniola, quickly, \ldots, \rangle$	2	RB
NNP	RB	$\langle Hispaniola, quickly, \ldots, \rangle$	3	VB
RB	VB	$\langle Hispaniola, quickly, \ldots, \rangle$	4	DT
VP	DT	$\langle Hispaniola, quickly, \ldots, \rangle$	5	JJ
DT	JJ	$\langle Hispaniola, quickly, \ldots, \rangle$	6	NN

A tagged sentence with n words has n history/tag pairs

Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/NN

History				Tag
t_{-2}	t_{-1}	$w_{[1:n]}$	i	t
*	*	$\langle Hispaniola, quickly, \ldots, \rangle$	1	NNP
*	NNP	$\langle Hispaniola, quickly, \ldots, \rangle$	2	RB
NNP	RB	$\langle Hispaniola, quickly, \ldots, \rangle$	3	VB
RB	VB	$\langle Hispaniola, quickly, \ldots, \rangle$	4	DT
VP	DT	$\langle Hispaniola, quickly, \ldots, \rangle$	5	JJ
DT	JJ	$\langle Hispaniola, quickly, \ldots, \rangle$	6	NN

Define global features through local features:

$$\mathbf{f}(t_{[1:n]}, w_{[1:n]}) = \sum_{i=1}^{n} g(h_i, t_i)$$

where t_i is the i'th tag, h_i is the i'th history

Global and Local Features

- Typically, local features are indicator functions, e.g.,

$$g_{101}(h,t) = \begin{cases} 1 & \text{if current word } w_i \text{ ends in } \texttt{ing} \text{ and } t = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

- and global features are then counts,

$f_{101}(w_{[1:n]}, t_{[1:n]})$ = Number of times a word ending in `ing` is tagged as `VBG` in $(w_{[1:n]}, t_{[1:n]})$

Putting it all Together

- $\mathbf{GEN}(w_{[1:n]})$ is the set of all tagged sequences of length n
- $\mathbf{GEN}$, $\mathbf{f}$, $\mathbf{v}$ define

$$
\begin{aligned}
F(w_{[1:n]}) &= \arg\max_{t_{[1:n]} \in \mathbf{GEN}(w_{[1:n]})} \mathbf{v} \cdot \mathbf{f}(w_{[1:n]}, t_{[1:n]}) \\
&= \arg\max_{t_{[1:n]} \in \mathbf{GEN}(w_{[1:n]})} \mathbf{v} \cdot \sum_{i=1}^{n} g(h_i, t_i) \\
&= \arg\max_{t_{[1:n]} \in \mathbf{GEN}(w_{[1:n]})} \sum_{i=1}^{n} \mathbf{v} \cdot g(h_i, t_i)
\end{aligned}
$$

Dynamic programming can be used to find the argmax!

A Variant of the Perceptron Algorithm

Inputs: Training set (x_i, y_i) for $i = 1 \ldots n$

Initialization: $\mathbf{v} = 0$

Define: $F(x) = \text{argmax}_{y \in \mathbf{GEN}(x)} \mathbf{f}(x, y) \cdot \mathbf{v}$

Algorithm: For $t = 1 \ldots T$, $i = 1 \ldots n$

$z_i = F(x_i)$

If $(z_i \neq y_i)$ $\quad \mathbf{v} = \mathbf{v} + \mathbf{f}(x_i, y_i) - \mathbf{f}(x_i, z_i)$

Output: Parameters $\mathbf{v}$

Training a Tagger Using the Perceptron Algorithm

Inputs: Training set $(w^i_{[1:n_i]}, t^i_{[1:n_i]})$ for $i = 1 \dots n$.

Initialization: $\mathbf{v} = 0$

Algorithm: For $t = 1 \dots T, i = 1 \dots n$

$$z_{[1:n_i]} = \arg \max_{u_{[1:n_i]} \in \mathcal{T}^{n_i}} \mathbf{v} \cdot \mathbf{f}(w^i_{[1:n_i]}, u_{[1:n_i]})$$

$z_{[1:n_i]}$ can be computed with the dynamic programming (Viterbi) algorithm

If $z_{[1:n_i]} \neq t^i_{[1:n_i]}$ then

$$\mathbf{v} = \mathbf{v} \quad + \quad \mathbf{f}(w^i_{[1:n_i]}, t^i_{[1:n_i]}) - \mathbf{f}(w^i_{[1:n_i]}, z_{[1:n_i]})$$

Output: Parameter vector $\mathbf{v}$.

An Example

Say the correct tags for i'th sentence are

the/DT man/NN bit/VBD the/DT dog/NN

Under current parameters, output is

the/DT man/NN bit/NN the/DT dog/NN

Assume also that features track: (1) all bigrams; (2) word/tag pairs

Parameters incremented:

$$\langle \text{NN, VBD} \rangle, \langle \text{VBD, DT} \rangle, \langle \text{VBD} \rightarrow \text{bit} \rangle$$

Parameters decremented:

$$\langle \text{NN, NN} \rangle, \langle \text{NN, DT} \rangle, \langle \text{NN} \rightarrow \text{bit} \rangle$$

Experiments

- Wall Street Journal part-of-speech tagging data

  Perceptron $=$ 2.89% error, Log-linear tagger $=$ 3.28% error

- [Ramshaw and Marcus, 1995] NP chunking data

  Perceptron $=$ 93.63% accuracy, Log-linear tagger $=$ 93.29% accuracy