Sequence Labeling

Grzegorz Chrupała and Nicolas Stroppa

Google
Saarland University

META

Outline

Entity recognition in news

West Indian all-rounder Phil Simons$_{\text{PERSON}}$ took four for 38 on Friday as Leicestershire ...

- We want to categorize news articles based on which entities they talk about
- We can annotate a number of articles with appropriate labels
- And learn a model from the annotated data
- Assigning labels to words in a sentence is an example of a sequence labeling task

Sequence labeling

Word	POS	Chunk	NE
West	NNP	B-NP	B-MISC
Indian	NNP	I-NP	I-MISC
all-rounder	NN	I-NP	O
Phil	NNP	I-NP	B-PER
Simons	NNP	I-NP	I-PER
took	VBD	B-VP	O
four	CD	B-NP	O
for	IN	B-PP	O
38	CD	B-NP	O
on	IN	B-PP	O
Friday	NNP	B-NP	O
as	IN	B-PP	O
Leicestershire	NNP	B-NP	B-ORG
beat	VBD	B-VP	O

Sequence labeling

- Assigning sequences of labels to sequences of some objects is a very common task (NLP, bioinformatics)
- In NLP
  - Speech recognition
  - POS tagging
  - chunking (shallow parsing)
  - named-entity recognition

- In general, learn a function $h : \Sigma^* \rightarrow \mathcal{L}^*$ to assign a sequence of labels from $\mathcal{L}$ to the sequence of input elements from Σ
- The most easily tractable case: each element of the input sequence receives one label:

$$h : \Sigma^n \rightarrow \mathcal{L}^n$$

- In cases where it does not naturally hold, such as chunking, we decompose the task so it is satisfied.
- IOB scheme: each element gets a label indicating if it is initial in chunk X (B-X), a non-initial in chunk X (I-X) or is outside of any chunk (O).

Local classifier

- The simplest approach to sequence labeling is to just use a regular classifier, and make a local decision for each word.
- Predictions for previous words can be used in predicting the current word
- This straightforward strategy can sometimes give surprisingly good results

Outline

HMM refresher

- HMMs – simplified models of the process generating the sequences of interest
- Observations generated by hidden states
  - Analogous to classes
  - Dependencies between states

Formally

- Sequence of observations $\mathbf{x} = x_1, x_2, \ldots, x_N$
- Corresponding hidden states $\mathbf{z} = z_1, z_2, \ldots, z_N$

$$\begin{aligned}\hat{\mathbf{z}} &= \operatorname*{argmax}_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}) \\ &= \operatorname*{argmax}_{\mathbf{z}} \frac{P(\mathbf{x}|\mathbf{z})P(\mathbf{z})}{\sum_{\mathbf{z}} P(\mathbf{x}|\mathbf{z})P(\mathbf{z})} \\ &= \operatorname*{argmax}_{\mathbf{z}} P(\mathbf{x}, \mathbf{z})\end{aligned}$$

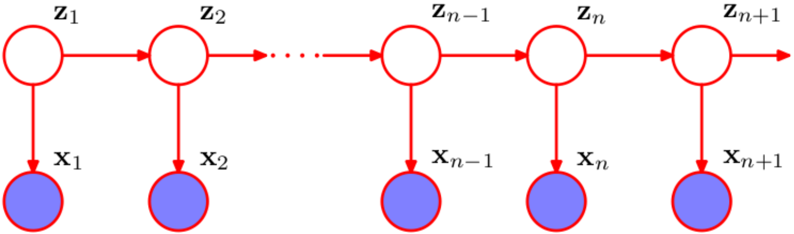
$$P(\mathbf{x}, \mathbf{z}) = \prod_{i=1}^{N} P(x_i | x_1, \ldots, x_{i-1}, z_1, \ldots, z_i) P(z_i | x_1, \ldots, x_{i-1}, z_1, \ldots, z_{i-1})$$

Simplifying assumptions

- Current state only depends on previous state
- Previous observation only influence current one via the state

$$P(x_1, x_2, \ldots, x_N, z_1, z_2, \ldots, z_N) = \prod_{i=1}^{N} P(x_i|z_i)P(z_i|z_{i-1})$$

- $P(x_i|z_i)$ – emission probabilities
- $P(z_i|z_{i-1})$ – transition probabilities



$\mathbf{z}_1$
$\mathbf{z}_2$
$\mathbf{z}_{n-1}$
$\mathbf{z}_n$
$\mathbf{z}_{n+1}$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_{n-1}$
$\mathbf{x}_n$
$\mathbf{x}_{n+1}$

A real Markov process

A dishonest casino

- A casino has two dice:
  - Fair die: $P(1) = P(2) = P(3) = P(5) = P(6) = 1/6$
  - Loaded die:
    $P(1) = P(2) = P(3) = P(5) = 1/10$
    $P(6) = 1/2$
- Casino player switches back-and-forth between fair and loaded die once every 20 turns on average

Evaluation question

- Given a sequence of rolls:
  124552646214614613613666166466163
  661636616361651561511514612356234 4
- How likely is this sequence, given our model of the casino?

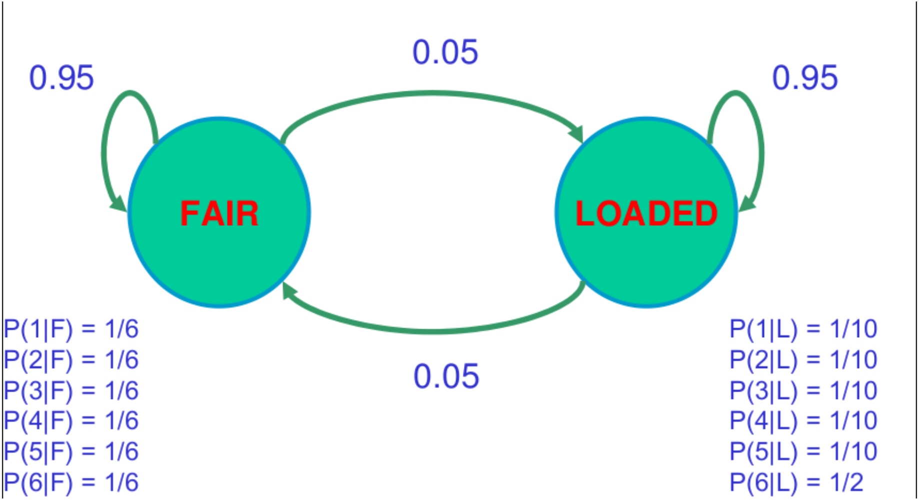
Decoding question

- Given a sequence of rolls:
  12455264621461461361366616646 6163
  6616366163616515615115146123562344
- Which throws were generated by the fair dice and which by the loaded dice?

Learning question

- Given a sequence of rolls:
  12455264621461461361366616646 6163
  66163661636165156151151461235623 44
- Can we infer how the casino works? How loaded is the dice? How often the casino player changes between the dice?

The dishonest casino model



Example

- Let the sequence of rolls be: $\mathbf{x} = (1, 2, 1, 5, 2, 1, 6, 2, 4)$
- A candidate parse is $\mathbf{z} = (F, F, F, F, F, F, F, F, F, F)$
- What is the probability $P(\mathbf{x}, \mathbf{z})$?

$$P(\mathbf{x}, \mathbf{z}) = \prod_{i=1}^{N} P(x_i | z_i) P(z_i | z_{i-1})$$

- (Let's assume initial transition probabilities $P(F|0) = P(L|0) = \frac{1}{2}$)

Example

- Let the sequence of rolls be: $\mathbf{x} = (1, 2, 1, 5, 2, 1, 6, 2, 4)$
- A candidate parse is $\mathbf{z} = (F, F, F, F, F, F, F, F, F, F)$
- What is the probability $P(\mathbf{x}, \mathbf{z})$?

$$P(\mathbf{x}, \mathbf{z}) = \prod_{i=1}^{N} P(x_i | z_i) P(z_i | z_{i-1})$$

- (Let's assume initial transition probabilities $P(F|0) = P(L|0) = \frac{1}{2}$)

$$\begin{aligned} & \frac{1}{2} \times P(1|F)P(F|F) \times P(2|F)P(F|F) \cdots P(4|L) \\ &= \frac{1}{2} \times \left(\frac{1}{6}\right)^{10} \times 0.95^{9} \\ &= 5.21 \times 10^{-9} \end{aligned}$$

Example

- Let the sequence of rolls be: $\mathbf{x} = (1, 2, 1, 5, 2, 1, 6, 2, 4)$
- A candidate parse is $\mathbf{z} = (F, F, F, F, F, F, F, F, F, F)$
- What is the probability $P(\mathbf{x}, \mathbf{z})$?

$$P(\mathbf{x}, \mathbf{z}) = \prod_{i=1}^{N} P(x_i|z_i)P(z_i|z_{i-1})$$

- (Let's assume initial transition probabilities $P(F|0) = P(L|0) = \frac{1}{2}$)

$$\frac{1}{2} \times P(1|F)P(F|F) \times P(2|F)P(F|F) \cdots P(4|L)$$
$$= \frac{1}{2} \times \left(\frac{1}{6}\right)^{10} \times 0.95^9$$
$$= 5.21 \times 10^{-9}$$

Example

- What about the parse $\mathbf{z} = (L, L, L, L, L, L, L, L, L, L)$?

$$\frac{1}{2} \times P(1|L)P(L|L) \times P(2|L)P(L|L) \cdots P(4|L)$$
$$= \frac{1}{2} \times 0.5^2 \times 0.95^0 = 7.9 \times 10^{-10}$$

- It's 6.61 times more likely that the all the throws came from a fair dice than that they came from a loaded dice.

Example

- Now let the throws be: $\mathbf{x} = (1, 6, 6, 5, 6, 2, 6, 6, 3, 6)$
- What is $P(\mathbf{x}, F^{10})$ now?

Example

- Now let the throws be: $\mathbf{x} = (1, 6, 6, 5, 6, 2, 6, 6, 3, 6)$
- What is $P(\mathbf{x}, F^{10})$ now?

$$\frac{1}{2} \times \left(\frac{1}{6}\right)^{10} \times 0.95^9 = 5.21 \times 10^{-9}$$

Same as before

Example

- Now let the throws be: $\mathbf{x} = (1, 6, 6, 5, 6, 2, 6, 6, 3, 6)$
- What is $P(\mathbf{x}, F^{10})$ now?

$$\frac{1}{2} \times \left(\frac{1}{6}\right)^{10} \times 0.95^9 = 5.21 \times 10^{-9}$$

Same as before

- What is $P(\mathbf{x}, L^{10})$

Example

- Now let the throws be: $\mathbf{x} = (1, 6, 6, 5, 6, 2, 6, 6, 3, 6)$
- What is $P(\mathbf{x}, F^{10})$ now?

$$\frac{1}{2} \times \left(\frac{1}{6}\right)^{10} \times 0.95^9 = 5.21 \times 10^{-9}$$

Same as before

- What is $P(\mathbf{x}, L^{10})$

$$\frac{1}{2} \times 0.1^4 \times 0.5^6 \times 0.95^9 = 0.5 \times 10^{-7}$$

- So now it is 100 times more likely that all the throws came from a loaded dice

Decoding

- Given $\mathbf{x}$ we want to find the best $\mathbf{z}$, i.e. the one which maximizes $P(\mathbf{x}, \mathbf{z})$

$$\hat{z} = \underset{\mathbf{z}}{\operatorname{argmax}}\, P(\mathbf{x}, \mathbf{z})$$

- Enumerate all possible $\mathbf{z}$, and evalue $P(\mathbf{x}, \mathbf{z})$?

Decoding

- Given $\mathbf{x}$ we want to find the best $\mathbf{z}$, i.e. the one which maximizes $P(\mathbf{x}, \mathbf{z})$

$$\hat{z} = \operatorname*{argmax}_{\mathbf{z}} P(\mathbf{x}, \mathbf{z})$$

- Enumerate all possible $\mathbf{z}$, and evalue $P(\mathbf{x}, \mathbf{z})$?
- Exponential in length of input

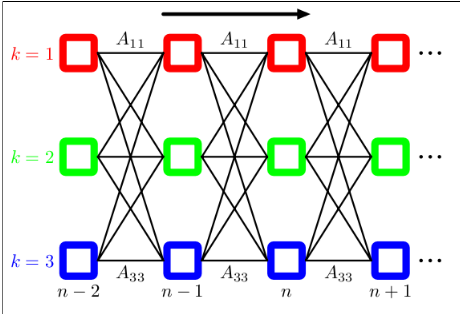
Decoding

- Given $\mathbf{x}$ we want to find the best $\mathbf{z}$, i.e. the one which maximizes $P(\mathbf{x}, \mathbf{z})$

$$\hat{z} = \operatorname*{argmax}_{\mathbf{z}} P(\mathbf{x}, \mathbf{z})$$

- Enumerate all possible $\mathbf{z}$, and evalue $P(\mathbf{x}, \mathbf{z})$?
- Exponential in length of input
- Dynamic programming to the rescue

Decoding



- Store intermediate results in a table for reuse
- Score to remember: probability of the most likely sequence of states up to position i, with state at position i being k

$$V_k(i) = \max_{z_1,\dots,z_{i-1}} P(x_1,\cdots,x_{i-1},z_1,\cdots,z_{i-1},x_i,z_i=k)$$

Decoding

- We can define $V_k(i)$ recursively

$$
\begin{aligned}
V_l(i+1) &= \max_{z_1,\ldots,z_i} P(x_1,\ldots,x_i,z_1,\ldots,z_i,x_{i+1},z_{i+1}=l) \\
&= \max_{z_1,\ldots,z_i} P(x_{i+1},z_{i+1}=l|x_1,\ldots,x_i,z_1,\ldots,z_i) \\
&\qquad \times P(x_1,\ldots,x_i,z_1,\ldots,z_i) \\
&= \max_{z_1,\ldots,z_i} P(x_{i+1},z_{i+1}=l|z_i)P(x_1,\ldots,x_i,z_1,\ldots,z_i) \\
&= \max_k P(x_{i+1},z_{i+1}=l) \max_{z_1,\ldots,z_{i-1}} P(x_1,\cdots,x_i,z_1,\cdots,z_i=k) \\
&= \max_k P(x_{i+1},z_{i+1}=l) V_k(i) \\
&= P(x_{i+1}|z_{i+1}=l) \max_k P(z_{i+1}=k|z_i=l) V_k(i)
\end{aligned}
$$

- We introduce simplified notation for the parameters

$$V_l(i+1) = E_l(x_{i+1}) \max_k A_{kl} V_k(i)$$

Viterbi algorithm

- Input $\mathbf{x} = (x_1, \ldots, x_N)$
- Initialization

$$V_0(0) = 1 \text{ where 0 is the fake starting position}$$
$$V_k(0) = 0 \text{ for all } k > 0$$

- Recursion

$$V_l(i) = E_l(x_i) \max_k A_{kl} V_k(i-1)$$
$$Z_l(i) = \operatorname*{argmax}_k A_{kl} V_k(i-1)$$

- Termination

$$P(\mathbf{x}, \hat{\mathbf{z}}) = \max_k V_k(N)$$
$$\hat{z_N} = \operatorname*{argmax}_k V_k(N)$$

Learning HMM

- Learning from labeled data

Learning HMM

- Learning from labeled data
  - Estimate parameters (emission and transition probabilities) from (smoothed) relative counts

Learning HMM

- Learning from labeled data
  - Estimate parameters (emission and transition probabilities) from (smoothed) relative counts

$$A_{kl} = \frac{C(k,l)}{\sum_{l'} C(k,l')}$$

$$E_k(x) = \frac{C(k,x)}{\sum_{x'} C(k,x')}$$

- Learning from unlabeled with Expectation Maximization
  - Start with randomly initialized parameters θ_0
  - Iterate until convergence
    - ★ Compute (soft) labeling given current θ_i
    - ★ Compute updated parameters θ_{i+1} from this labeling

Outline

Maximum Entropy Markov Models

- Model structure like in HMM
- Logistic regression (Maxent) to learn $P(z_i|\mathbf{x}, z_{i-1})$
- For decoding, use learned probabilities and run Viterbi

HMMs and MEMMs

- HMM POS tagging model:

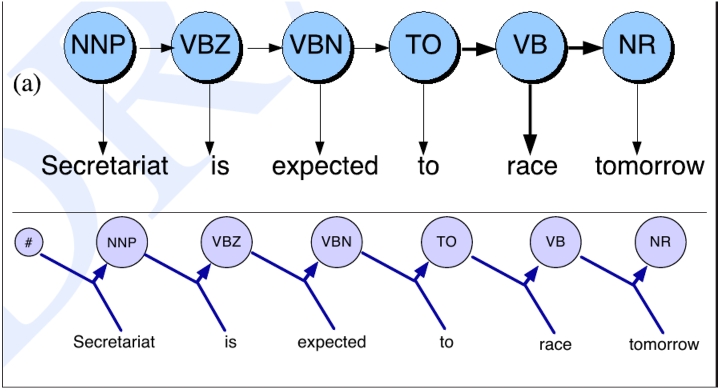
$$\hat{\mathbf{z}} = \operatorname*{argmax}_{\mathbf{z}} P(\mathbf{z}|\mathbf{x})$$
$$= \operatorname*{argmax}_{\mathbf{z}} P(\mathbf{x}|\mathbf{z})P(\mathbf{z})$$
$$= \operatorname*{argmax}_{\mathbf{z}} \prod_i P(x_i|z_i)P(z_i|z_{i-1})$$

- MEMM POS tagging model:

$$\hat{\mathbf{z}} = \operatorname*{argmax}_{\mathbf{z}} P(\mathbf{z}|\mathbf{x})$$
$$= \operatorname*{argmax}_{\mathbf{z}} \prod_i P(z_i|\mathbf{x}, z_{i-1})$$

- Maximum entropy model gives conditional probabilities

Conditioning probabilities in a HMM and a MEMM



Viterbi in MEMMs

- Decoding works almost the same as in HMM
- Except entries in the DP table are values of $P(z_i|\mathbf{x}, z_{i-1})$
- Recursive step: Viterbi value of time t for state j:

$$V_l(i+1) = \max_k P(z_{i+1} = l|\mathbf{x}, z_i = k) V_k(i)$$

Outline

Perceptron for sequences

$\textsc{SequencePerceptron}(\{\mathbf{x}\}^{1:N}, \{\mathbf{z}\}^{1:N}, I)$:

$\mathbf{w} \leftarrow \mathbf{0}$
for $i = 1 \ldots I$ **do**
  for $n = 1 \ldots N$ **do**
    $\hat{\mathbf{y}}^{(n)} \leftarrow \mathrm{argmax}_{\mathbf{z}}\, \mathbf{w} \cdot \Phi(\mathbf{x}^{(n)}, \mathbf{z})$
    if $\hat{\mathbf{z}}^{(n)} \neq \mathbf{z}^{(n)}$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + \Phi(\mathbf{x}^{(n)}, \mathbf{z}^{(n)}) - \Phi(\mathbf{x}^{(n)}, \hat{\mathbf{z}}^{(n)})$
return w

Feature function

$\text{Harry}_{\text{PER}}\ \text{loves}_{\text{O}}\ \text{Mary}_{\text{PER}}$

$$\Phi(\mathbf{x}, \mathbf{z}) = \sum_i \phi(\mathbf{x}, z_{i-1}, z_i)$$

i	$x_i = \text{Harry} \wedge z_i = \text{PER}$	$\text{suff}_2(x_i) = \text{ry} \wedge z_i = \text{PER}$	$x_i = \text{loves} \wedge z_i = \text{O}$
1	1	1	0
2	0	0	1
3	0	1	0
Φ	1	2	1

Search

$$\hat{\mathbf{z}}^{(n)} = \operatorname*{argmax}_{\mathbf{z}} \mathbf{w} \cdot \Phi(\mathbf{x}^{(n)}, \mathbf{z})$$

Global score is computed incrementally:

$$\mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{|\mathbf{x}|} \mathbf{w} \cdot \phi(\mathbf{x}, z_{i-1}, z_i)$$

Update term

$$\mathbf{w}^{(n)} = \mathbf{w}^{(n-1)} + \left[\Phi(\mathbf{x}^{(n)}, \mathbf{z}^{(n)}) - \Phi(\mathbf{x}^{(n)}, \hat{\mathbf{z}}^{(n)})\right]$$

$$\begin{aligned} & \Phi(\text{Harry loves Mary}, \text{PER O PER}) \\ - \quad & \Phi(\text{Harry loves Mary}, \text{ORG O PER}) = \end{aligned}$$

$x_i = \text{Harry} \wedge z_i = \text{PER}$	$x_i = \text{Harry} \wedge z_i = \text{ORG}$	$\text{suff}_2(x_i) = \text{ry} \wedge z_i = \text{PER}$	$\cdots$
1	0	2	$\cdots$
0	1	1	$\cdots$
1	-1	1	$\cdots$

Comparison

Model	HMM	MEMM	Perceptron
Type	Generative	Discriminative	Discriminative
Distribution	$P(\mathbf{x}, \mathbf{z})$	$P(\mathbf{z}\|\mathbf{x})$	N/A
Smoothing	Crucial	Optional	Optional
Output dep.	Chain	Chain	Chain
Sup. learning	No decoding	No decoding	With decoding

The end