RITAL

Information retrieval and natural language processing
Recherche d'information et traitement automatique de la langue

Master 1 DAC, semestre 2

Nicolas Thome

SCIENCES SORBONNE UNIVERSITÉ

ISIR INSTITUT DES SYSTÈMES INTELLIGENTS ET DE ROBOTIQUE

MLIA Machine Learning & Deep Learning for Information Access

Sequence Processing

Outline

- BoW and unsupervised methods (LSA/LDA): no explicit representation of words' sequences
- An important feature for many NLP task

$\Rightarrow$ How to include such sequential nature?

- Markov Models (MM) and HMM: generative models
- Conditional Random Fields (CRF): discriminative models $\rightarrow$ today
- RNNs & transformers: next course

General framework

- Input $\boldsymbol{x} \in \mathcal{X}$: arbitrary
- Output $y \in \mathcal{Y}$: discrete output space
  - $\boldsymbol{y} \in \mathcal{Y}$ can be anything: vector, sequence (of vectors), graph, etc
- Assumption, there exist a function $\psi(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^d$
- CRF: model of conditional probability function:

$$P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) = \frac{e^{\boldsymbol{w}^T \psi(\boldsymbol{x}, \boldsymbol{y})}}{\sum_{y' \in \mathcal{Y}} e^{\boldsymbol{w}^T \psi(\boldsymbol{x}, \boldsymbol{y}')}}$$

- $Z(\boldsymbol{x}) = \sum_{y' \in \mathcal{Y}} e^{w^T \psi(\boldsymbol{x}, \boldsymbol{y}')}$: partition function

SCIENCES SORBONNE UNIVERSITÉ

General framework for training

- A set of N training pairs $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i \in \{1;N\}}$
- i.i.d samples, maximizing conditional log likelihood $\mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{N} log\left[P(\boldsymbol{y}_i|\boldsymbol{x}_i, \boldsymbol{w})\right]$

$$\mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{N} w^T \psi(\boldsymbol{x}_i, \boldsymbol{y}_i) - log\left(\sum_{y' \in \mathcal{Y}} e^{\boldsymbol{w}^T \psi(\boldsymbol{x}_i, \boldsymbol{y}')}\right) = w^T \psi(\boldsymbol{x}_i, \boldsymbol{y}_i) - log\left(Z(\boldsymbol{x}_i)\right) \quad (1)$$

- $\boldsymbol{y}_i$ GT output for input $\boldsymbol{x}_i$
- Eq (1) convex, can be solved with gradient descent
  - Can add ℓ_2 regularization $||\boldsymbol{w}||^2$on Eq (1) by putting Gaussian prior $P(\boldsymbol{w}) \sim \mathcal{N}(0, \sigma^2)$

Classification

- We need to define $\mathcal{X}, \mathcal{Y}, \psi(\boldsymbol{x}, \boldsymbol{y})$
- Ex: $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{1; K\}$ (K number of classes), and $\psi(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^{d \cdot K}$

$$\psi(\boldsymbol{x}, \boldsymbol{y}) = (\delta_{y,1}\boldsymbol{x}, ...\delta_{y,k}\boldsymbol{x}, ...\delta_{y,K}\boldsymbol{x})$$

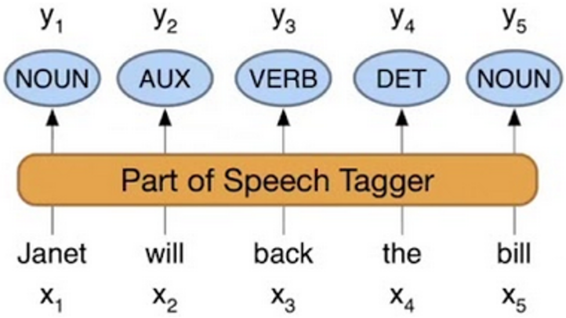
- $\boldsymbol{w} = (\boldsymbol{w}_1, ...\boldsymbol{w}_k, ...\boldsymbol{w}_K)$, where $\boldsymbol{w}_k \in \mathbb{R}^d \rightarrow \boldsymbol{w}^T\psi(\boldsymbol{x}, k) = \boldsymbol{w}_k^T\boldsymbol{x}$
- Training: $\{(\boldsymbol{x}_i, k_i^*)\}_{i \in \{1;N\}}$

$$\Rightarrow P(k|\boldsymbol{x}, \boldsymbol{w}) = \frac{e^{\boldsymbol{w}_k^T\boldsymbol{x}}}{\sum\limits_{k'=1}^{K} e^{\boldsymbol{w}_{k'}^T\boldsymbol{x}}}, \text{ and } \mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{N} \boldsymbol{w}_{k_i^*}^T\boldsymbol{x}_i - log\left(\sum_{k=1}^{K} e^{\boldsymbol{w}_k^T\boldsymbol{x}_i}\right)$$

$\boxed{\textbf{Logistic Regression}}$

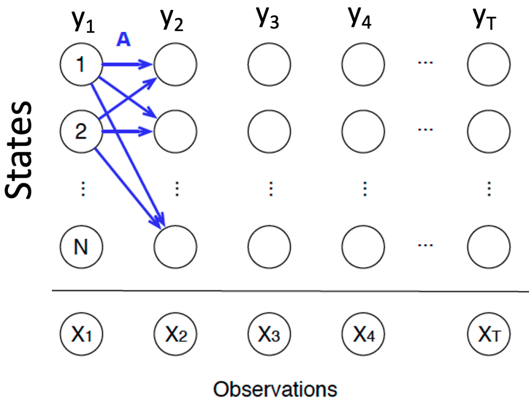
Sequences: ex for Part of Speech tagging

- Context: labelling of each word $\rightarrow$ Part of Speech (PoS)
- Input $\boldsymbol{x} = (x_1, ...x_T)$ a sentence with T words
- Output: $\boldsymbol{y} = (y_1, ...y_T)$ a sequence of T PoS



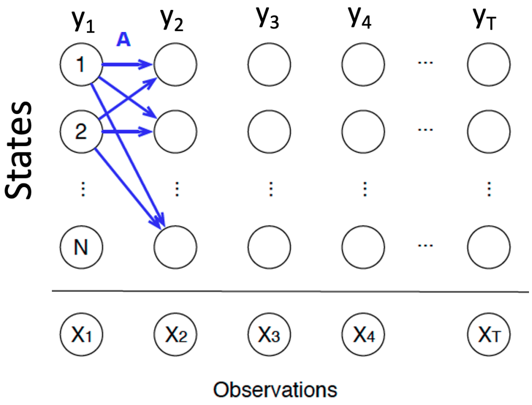
HMMs

- Input $\boldsymbol{x} = (x_1, ...x_T)$ a sentence with T words
  - K words, e.g. $K \sim 20000$
- Output: $\boldsymbol{y} = (y_1, ...y_T)$ a sequence of T PoS
  - N PoS tags, e.g. $N = 20$
- HMM parameters: observation matrix $\mathrm{B} \in \mathbb{R}^{K \cdot N}$, transition matrix $\mathrm{A} \in \mathbb{R}^{N \cdot N}$
- Training: states observed (not hidden) $\rightarrow$ A, B counting!
- Predicting PoS tags for a new sequence $\rightarrow$ Viterbi (see MAPSI)

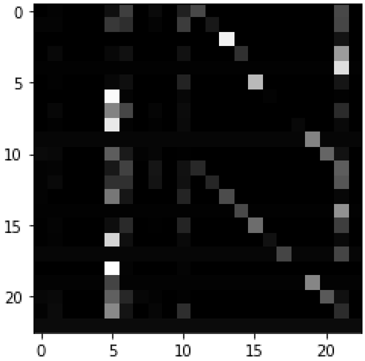


HMMs

- Input $\boldsymbol{x} = (x_1, ...x_T)$ a sentence with T words
  - K words, e.g. $K \sim 20000$
- Output: $\boldsymbol{y} = (y_1, ...y_T)$ a sequence of T PoS
  - N PoS tags, e.g. $N = 20$
- HMM parameters: observation matrix $B \in \mathbb{R}^{K \cdot N}$, transition matrix $A \in \mathbb{R}^{N \cdot N}$
- Training: states observed (not hidden) $\rightarrow$ A, B counting!
- Predicting PoS tags for a new sequence $\rightarrow$ Viterbi (see MAPSI)



- Some words are ambiguous: may be to several PoS tags
- Sequence order: help to resolve some ambiguities



Examples of features for PoS Tagging

```
def features_full(sentence, index):
    return {
        'word': sentence[index],
        'is_first': index == 0,
        'is_last': index == len(sentence) - 1,
        'is_capitalized': sentence[index][0].upper() == sentence[index][0],
        'is_all_caps': sentence[index].upper() == sentence[index],
        'is_all_lower': sentence[index].lower() == sentence[index],
        'prefix-1': sentence[index][0],
        'prefix-2': sentence[index][:2],
        'prefix-3': sentence[index][:3],
        'suffix-1': sentence[index][-1],
        'suffix-2': sentence[index][-2:],
        'suffix-3': sentence[index][-3:],
        'prev_word': '' if index == 0 else sentence[index - 1],
        'next_word': '' if index == len(sentence) - 1 else sentence[index + 1],
        'has_hyphen': '-' in sentence[index],
        'is_numeric': sentence[index].isdigit(),
        'capitals_inside': sentence[index][1:].lower() != sentence[index][1:]
    }
```

$$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{e^{\sum_{t=1}^{T}\sum_{k=1}^{K}[\theta_k u_k(y_t,\boldsymbol{x}) + \lambda_k p_k(y_{t-1},y_t,\boldsymbol{x})]}}{Z(\boldsymbol{x})}$$

$$Z(\boldsymbol{x}) = \sum_{y' \in \mathcal{Y}} e^{\sum_{t=1}^{T}\sum_{k=1}^{K}\left[\theta_k u_k(y'_t,\boldsymbol{x}) + \lambda_k p_k(y'_{t-1},y'_t,\boldsymbol{x})\right]} \tag{2}$$

- Problem: $\mathcal{Y}$ huge: dim N^T
- Model prediction: $\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x})$
- Training : computing normalization factor $Z(\boldsymbol{x}) = \sum_{y' \in \mathcal{Y}} \cdots$
- Brute-force computation intractable, must exploit structure...

$\Rightarrow$ **dynamic programming**

Inference

- Model prediction:

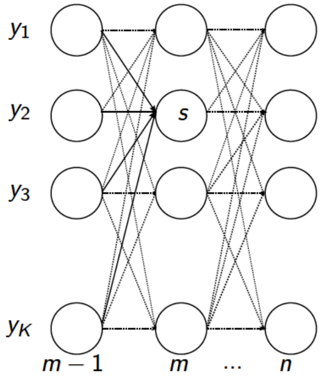
$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x})$$

$$= \arg\max_{\boldsymbol{y}} \sum_{t=1}^{T} \sum_{k=1}^{K} F_k(y_{t-1}, y_t, \boldsymbol{x})$$

$$:= \arg\max_{\boldsymbol{y}} \sum_{t=1}^{T} g_t(y_{t-1}, y_t)$$

- Viterbi algorithm $\sim$ HMM

$$\delta_m(s) \triangleq \max_{\{y_1,\cdots,y_{m-1}\}} \left[\sum_{i=1}^{m-1} g_i(y_{i-1}, y_i) + g_m(y_{m-1}, s) \right]$$



Training

$$\mathcal{L}(\boldsymbol{w}) = w^T \psi(\boldsymbol{x}_i, \boldsymbol{y}_i) - log\left(Z(\boldsymbol{x}_i)\right)$$

- $Z(\boldsymbol{x}_i)$ intractable
- Forward-backward algorithm

Algorithm 1 The forward-backward algorithm for the probability calculation of the CRF

Input: The model $P(Y|X)$, the input sequence x, the output sequence y, and the location i

Output: The conditional probabilities $P(Y_i = y_i|x)$, $P(Y_{i-1} = y_{i-1}, Y_i = y_i|x)$

1. Let $M_i(y_{i-1}, y_i|x) = \exp(\sum_{i,k} \lambda_k t_k(y_{i-1}, y_i, x, i) + \sum_{i,j} \mu_j s_j(y_i, x, i))$, $y_0 = start$, $y_{n+1} = stop$
2. Initialization, $\alpha_0(y_0|x) = 1$, $\beta_{n+1}(y_{n+1}|x) = 1$
3. Recursion, for $k = 1, 2, \ldots, i$
   $\alpha_k^T(y_k|x) = \alpha_{k-1}^T(y_{k-1}|x) M(y_{k-1}, y_k|x)$
4. Recursion, for $j = n, n-1, \ldots, i+1, i, i-1, \ldots, 1$
   $\beta_j(y_j|x) = M_{j+1}(y_j, y_{j+1}|x) \beta_{j+1}(y_{j+1}|x)$
5. Calculation, $Z(x) = 1^T \times \beta_1(x)$
6. Calculation, $P(Y_i = y_i|x) = \alpha_i^T(y_i|x) \beta_i(y_i|x) / Z(x)$
   $P(Y_{i-1} = y_{i-1}, Y_i = y_i|x) = \alpha_{i-1}^T(y_{i-1}|x) M_i(y_{i-1}, y_i|x) \beta_i(y_i|x) / Z(x)$

CRF: conclusion

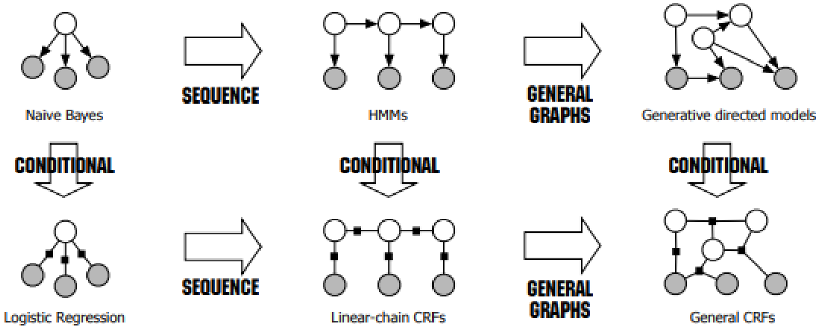


Fig. 2.3 Diagram of the relationship between naive Bayes, logistic regression, HMMs, linear-chain CRFs, generative models, and general CRFs.

Word Embeddings

Outline

1. Sequence Processing
2. Word Embeddings