Introduction to Conditional Random Fields

Jingbo Xia

Huazhong Agricultural University

xiajingbo.math@gmail.com

2020 年 11 月 18 日

Table of contents I

Outline

Roadmap and External Resources for Students

Roadmap for self study students

1) Textbook reading.
   i) Chapter 8. Graphical model. Christopher Bishop, 2006 [1] .
   ii) CRF. Roman K and Katrin T, 2007 [2].
2) Go through this slides.
3) Suggested Video. https://www.bilibili.com/video/av34444816?from=search&seid=13204054963054371637
4) Try "Tutorial on ME": https://github.com/bionlp-hzau/MaximumEntroy4Classification
5) Try "Tutorial on CRF": https://github.com/bionlp-hzau/Tutorial_4_CRF
   - ☐ Tools used in the tutorial: Wapiti, a python wrapper for fast linear-chain CRF.
6) More paper reading or Python your own. (Optional)

[1]Bishop, Christopher M. Pattern recognition and machine learning. springer, 2006.

[2]Klinger, Roman, and Katrin Tomanek. Classical probabilistic models and conditional random fields. TU, Algorithm Engineering, 2007.

Roadmap and External Resources for Students

External resources for students

1) Course Repo: https://github.com/bionlp-hzau/Tutorial_4_CRF
   i) Find slides there...
   ii) Find literature there...
   iii) Find tutorial there...
2) Slides: Kai-wei Chang's slides for forward and Viterbi on HMM [3].
3) More paper reading:
   1. First CRF paer. Lafferty, McCallum and Pereira, 2001. [4] .
   2. CRF paper. Charles Sutton and Andrew McCallum [5] .
   3. HMM Paper. Lawrence Rabiner, 1989. [6].
   4. Wikipages. https://en.wikipedia.org/wiki/Conditional_random_field

[3] http://www.cs.virginia.edu/~kc2wc/teaching/NLP16/slides/11-Viterbi.pdf

[4] Lafferty, John, Andrew McCallum, and Fernando CN Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data." (2001).

[5] Charles S, and McCallum A. "1 An Introduction to Conditional Random Fields for Relational Learning."https://publist.ist.ac.at/attachments/0000/0292/crf-tutorial.pdf

[6] Rabiner, Lawrence R. "A tutorial on hidden Markov models and selected applications in speech recognition." Proceedings of the IEEE 77, no. 2 (1989): 257-286.

Roadmap and External Resources for Students

More suggestions to students

More suggestions to students:

Suggestion 1: To understand the relationship among NB, HMM, ME and CRF looks fairly COOL.

1. Finish reading Roman and Katrin 2007.
2. An 90 minutes sketch story of NB, ME and CRF in a taste of "graphical model/conditional dependence assumption" is vividly introduced in bilibili video, as suggested in the previous page.
3. Meanwhile, a wrap-up sketch of Roman and Katrin's idea is shown in page 21.
   "I drew it!"

Suggestion 2: Hold on. To know the computation of CRF model is extremely COOL. So,

1. understand Viterbi on HMM first,
2. then make way to Viterbi on CRF,
3. ask yourself why Viterbi is not enough to solve CRF.

Outline

Fundamentals of Graphical Model I

As graphical model is a basis for understanding series of algorithms including CRF, Bishop's book [7] is strongly suggested. Please make sure to understand the following notations or methods.

i) Understand basic notations. For random variables $\boldsymbol{x} = \{x_1, \cdots, x_N\}$,
边缘概率 $P(x_i)$，条件概率 $P(x_i|x_j)$，联合概率 $P(x_1, x_2, \cdots, x_N)$，

Sum Rule: $P(x_i) = \int P(x_i, x_j)\,dx_j$, $(P(x_i) = \sum_{x_j} P(x_i, x_j))$

Product Rule: $P(x_i, x_j) = P(x_i)P(x_j|x_i)$

Chain Rule: $P(x_1, x_2, \cdots, x_p) = \prod_{n=1}^{N} P(x_n|x_1, x_2, \cdots, x_{n-1})$

Bayesian Rule: $P(x_j|x_i) = \frac{P(x_i,x_j)}{P(x_i)} = \frac{P(x_i,x_j)}{\int P(x_i,x_j)\,dx_j}$.

Fundamentals of Graphical Model II

ii) Understand conditional independence in **Bayesian Network**, aka, **directed graphical models**,
Define

$$a \perp\!\!\!\perp b | c \tag{1}$$

as the short form for "a, b are conditionally independence on c"". In another word,

$$P(a, b|c) = P(a|c)P(b|c),$$

or

$$P(a|b, c) = P(a|c).$$

(Note: We have $P(a, b|c) = P(a|b, c)P(b|c) = P(a|c)P(b|c)$.)

Fundamentals of Graphical Model III

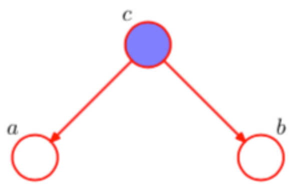
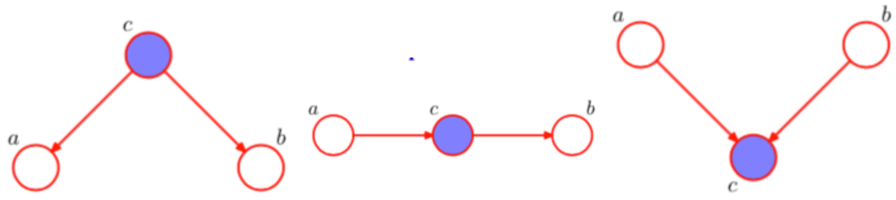


图 1: a, b conditioned on the value of variable c (shaded circle). Arrows starts from the variables c on which the distribution is conditioned.

The figure depicts the fact $a \perp\!\!\!\perp b | c$, or in another word,
$P(a, b|c) = P(a|c)P(b|c)$.
"The node c is said to be tail-to-tail with respect to this path because the node is connected to the tails of the two arrows, and the presence of such a path connecting nodes a and b causes these nodes to be dependent. However, when we condition on node c, as in Figure 1, the conditioned node 'blocks' the path from a to b and causes a and b to become (conditionally) independent."
—This come out an easy-to-follow illustration.

Fundamentals of Graphical Model IV

iii) Understand D-separation/“tail-to-tail’,“head-to-tail”,“head-to-head”;



定理 (D-Separation)

$A \perp\!\!\!\perp B|C$ is implied by a given directed acyclic graph (DAG). To do so, we consider all possible paths from any node in A to any node in B. Any such path is said to be blocked if it includes a node such that either

(a) *the arrows on the path meet either head-to-tail or tail-to-tail at the node, and the node is in the set C,*

or

(b) *the arrows meet head-to-head at the node, and neither the node, nor any of its descendants, is in the set C.*

Fundamentals of Graphical Model VI

iv) Understand conditional independence in **Markov Random Field**, aka, **Undirected graphical models**. (Unlike Bayesian network, we do not assume knowing how variables are conditioned, instead, the conditioning might be arbitrary, even mutual. Thus, the arrow is missing in the graph, and the D-separation is not applied in this circumstance, of coz.) Still, one need to start the discussion by define conditional independence as well.

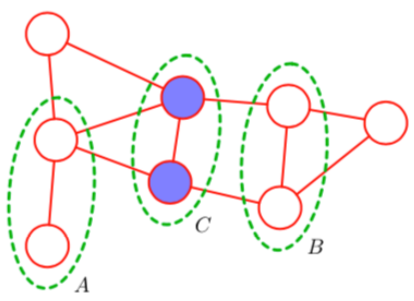


图 3: Paths from A to B are 'blocked' and so A, B is conditional independent on the condition of C.

Fundamentals of Graphical Model VII

We consider all possible paths that connect nodes in set A to nodes in set B. If all such paths pass through one or more nodes in set C, then all such paths are ‘blocked’ and so the conditional independence property holds:

$$A \perp\!\!\!\perp B | C.$$

v) Understand $\mathcal{C}$ Clique in undirectional graph;
A Clique is a good structure in which all the nodes are fully connected. In that way, it is impossible to 'block' each pair of variables, and make them conditional dependence. Finally, it is natural to define a pdf to a whole clique, instead of 'd-separating' them.
Clique: There exists a link between all pairs of nodes in a clique $\mathcal{C}$. In other words, the set of nodes in a clique is fully connected.
Maximal clique: a maximal clique is a clique such that it is not possible to include any other nodes from the graph in the set without it ceasing to be a clique.

Fundamentals of Graphical Model VIII

Let us denote all the random variables by $\boldsymbol{x}$, the set of variables in that clique by $\boldsymbol{x}_{\mathcal{C}}$, then the joint distribution is written as a product of *potential functions* 势函数 $\Psi_{\mathcal{C}}(\boldsymbol{x}_{\mathcal{C}})$ over the maximal cliques of the graph:

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{\mathcal{C}} \Psi_{\mathcal{C}}(\boldsymbol{x}_{\mathcal{C}}), \tag{2}$$

here the quantity Z, sometimes called the partition function, is a normalization constant and is given by $Z = \sum_{\boldsymbol{x}} \prod_{\mathcal{C}} \Psi_{\mathcal{C}}(\boldsymbol{x}_{\mathcal{C}})$.

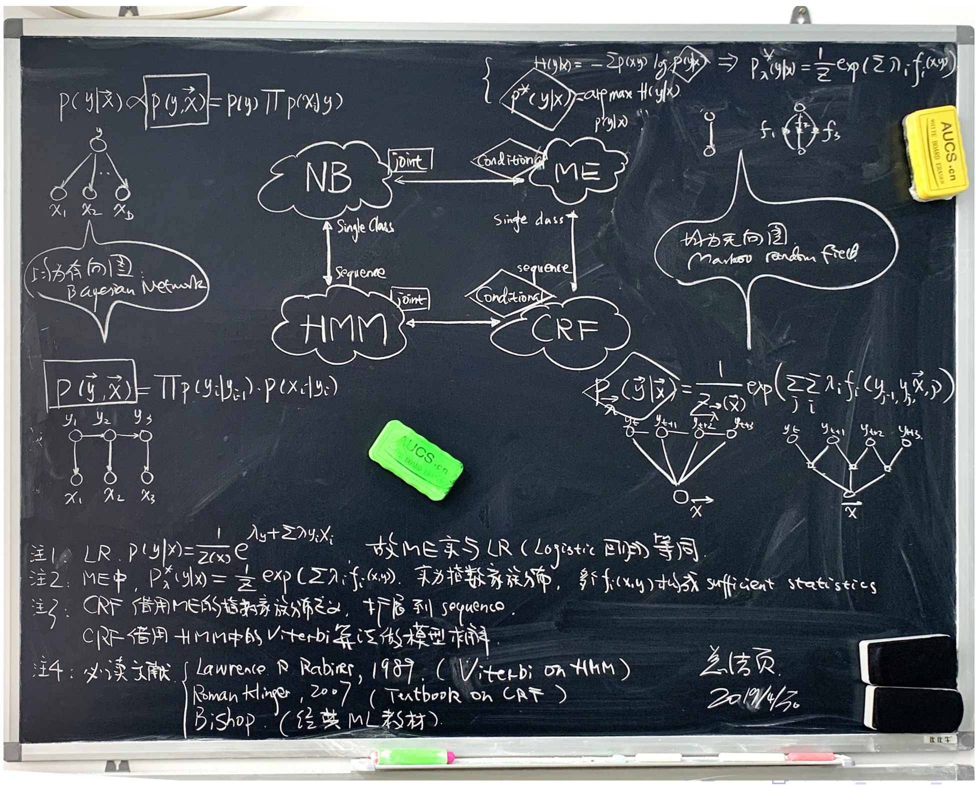
vi) Understand factor graph;

vii) Generative model for modeling $P(X, Y)$; Discriminative model for modeling $P(Y|X)$.

[7] Bishop, Christopher M. Pattern recognition and machine learning. springer, 2006.

Outline

From Naive Bayes, HMM, ME to CRF



$p(y|\vec{x}) \propto \boxed{p(y,\vec{x})} = p(y)\prod p(x_j|y)$
y
x_1 x_2 x_D
均为有向图 Bayesian Network
$\boxed{p(\vec{y},\vec{x})} = \prod p(y_i|y_{i-1})\cdot p(x_i|y_i)$
y_1 y_2 y_3
x_1 x_2 x_3
NB
joint
Conditional
ME
Single Class
sequence
Single class
sequence
HMM
joint
Conditional
CRF
$H(y|x) = -\sum p(x,y)\lg p(y|x) \Rightarrow P_\lambda^*(y|x) = \frac{1}{Z}\exp(\sum\lambda_i f_i(x,y))$
$p^*(y|x) = \arg\max_{p(y|x)} H(y|x)$
f_1 f_2 f_3
均为无向图 Markov random field
$P_{\vec{\lambda}}(\vec{y}|\vec{x}) = \frac{1}{Z_{\vec{\lambda}}(\vec{x})}\exp\left(\sum_j\sum_i \lambda_i f_i(y_{j-1},y_j,\vec{x},j)\right)$
y_t y_{t+1} y_{t+2} y_{t+3}
$\vec{x}$
y_t y_{t+1} y_{t+2} y_{t+3}
$\vec{x}$

注1: LR. $p(y|x)=\frac{1}{Z(x)}e^{\lambda_y+\sum\lambda_{y_i}x_i}$ 故ME实与LR (Logistic 回归) 等同.

注2: ME中, $P_\lambda^*(y|x)=\frac{1}{Z}\exp(\sum\lambda_i f_i(x,y))$. 实为指数家族分布, 其 $f_i(x,y)$ 构成 sufficient statistics

注3: CRF 借用ME的指数家族分布定义, 扩展到 sequence.
CRF借用HMM中的Viterbi算法做模型求解.

注4: 必读文献 { Lawrence R Rabiner, 1989. (Viterbi on HMM)
Roman Klinger, 2007 (Textbook on CRF)
Bishop. (经典ML教材).

[illegible]
2019/4/30.

From Naive Bayes, HMM, ME to CRF I

NB, HMM vs. ME, CRF, Bayesian network vs. Markov random field

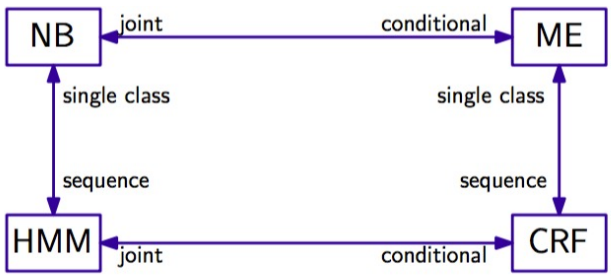


图 5: Relations among NB, HMM, ME and CRF algorithms. (from Roman and Katrin)

Note 1: Please double check the logic shown in page 21; Please as well check how linear chain CRF is developed on the basis of ME and HMM:

1. CRF has very similar structure in graph with HMM. Only difference: directional vs undirectional.

From Naive Bayes, HMM, ME to CRF II

NB, HMM vs. ME, CRF, Bayesian network vs. Markov random field

2. ME contributes the idea of formation of potential function (via feature function), $\exp\left(\sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}_C)\right)$. Linear chain CRF has cliques, i.e., $\boldsymbol{x}_C = \{y_{i-1}, y_i, x_{1:N}\}$.

Note 2: Both NB and HMM model the joint prob. They correspond to generative models.
Naive Bayes:

$$p^*(y|\vec{x}) = \underset{p(y|\vec{x})}{\mathrm{argmax}} p(y|\vec{x}), \quad \text{where } p(y|\vec{x}) \propto p(y,\vec{x}) = p(y)\prod_{i=1}^{m} p(x_i|y). \tag{3}$$

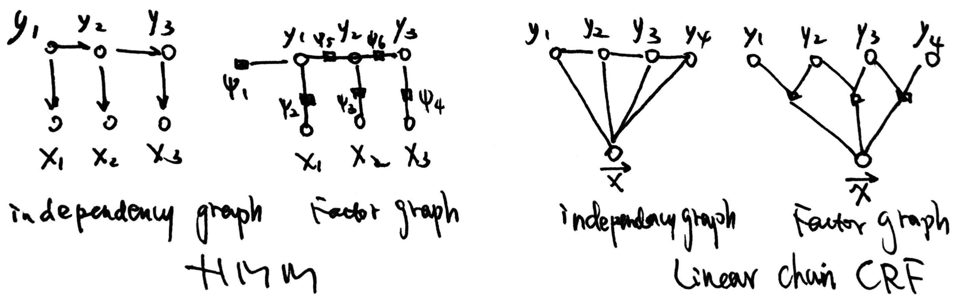
Similarly, HMM:

$$p(\vec{y},\vec{x}) = \prod_{i=0}^{n} p(y_i|y_{i-1})\, p(x_i|y_i). \tag{4}$$

From Naive Bayes, HMM, ME to CRF III

NB, HMM vs. ME, CRF, Bayesian network vs. Markov random field

Note 3: From HMM to CRF, we convert directional graphical model to undirectional graphical model. Moreover, we convert local normalization to global normalization and avoid label bias problem, see Lafferty et al 2001 [8]



[8] Lafferty, John, Andrew McCallum, and Fernando CN Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data." (2001).

From Naive Bayes, HMM, ME to CRF I

ME !

ME [9] [10] [11]:

$$p^*(y|\vec{x}) = \underset{p(y|\vec{x})}{\mathrm{argmax}} H(Y|X), \tag{5}$$

where the entropy

$$H(Y|X) = -\sum_{\vec{x}\in\mathcal{X}, y\in\mathcal{Y}} p(\vec{x}, y)\log p(y|\vec{x})$$

could be viewed as

$$H(Y|X) = \mathbb{E}(f(X, Y)),$$

where $f: (\vec{x}, y) \to -\log(p(y|\vec{x}))$, which describe the the amount of information needed to describe the event $(Y = y)$ given $(X = \vec{x})$;
or weighted sum of $H(Y|X = \vec{x})$ for each possible value of $\vec{x}$, using $p(\vec{x})$ as the weights

$$H(Y|X) = \sum_{\vec{x}\in\mathcal{X}} p(\vec{x}) H(Y|X = \vec{x})$$

From Naive Bayes, HMM, ME to CRF II

ME !

Lagrangian function in ME:

$$\Lambda(p,\vec{\lambda}) = \underbrace{H(Y|X)}_{\text{Primal problem}} + \sum_{i=1}^{m} \lambda_i \underbrace{\left(E(f_i) - \tilde{E}(f_i)\right)}_{\substack{=0\\ \text{Constraints}}} + \lambda_{m+1} \underbrace{\left(\sum_{y\in\mathcal{Y}} p(y|\vec{x}) - 1\right)}_{\substack{=0\\ \text{Constraints}}} \quad (6)$$

定理 (pdf of p(y|x)with ME)

Solving equation (6) leads to an exponential family distribution for pdf of $p(y|x)$,

$$p^*_{\vec{\lambda}}(y|\vec{x}) = \frac{1}{Z_{\vec{\lambda}}(\vec{x})} \exp\left(\sum_{i=1}^{m} \lambda_i f_i(\vec{x},y)\right), \quad (7)$$

where $Z_{\vec{\lambda}}(\vec{x}) = \sum_{y\in\mathcal{Y}} \exp\left(\sum_{i=1}^{m} \lambda_i f_i(\vec{x},y)\right)$, *and makes it ready for the extension to CRF.*

From Naive Bayes, HMM, ME to CRF III

ME !

Proof: We compute the derivative of (6) over $p(y|\vec{x})$ separately.

(i) $\frac{\partial}{\partial p(y|\vec{x})} H(y|\vec{x}) = -\tilde{p}(\vec{x})(\log p(y|\vec{x}) + 1)$.

(ii) $E(f_i) = \sum_{(\vec{x},y)\in\mathcal{X}\times\mathcal{Y}} p(\vec{x},y) f_i(\vec{x},y) \approx \sum_{(\vec{x},y)\in\mathcal{X}_{Train}\times\mathcal{Y}} \tilde{p}(\vec{x}) p(y|\vec{x}) f_i(\vec{x},y)$,

$\tilde{E}(f_i) = \sum_{(\vec{x},y)\in\mathcal{X}_{Train}\times\mathcal{Y}} \tilde{p}(\vec{x},y) f_i(\vec{x},y)$. So,

$\frac{\partial}{\partial p(y|\vec{x})} \sum_{i=1}^{m} \lambda_i \left(E(f_i) - \tilde{E}(f_i)\right) = \sum_{i=1}^{m} \lambda_i \tilde{p}(\vec{x}) f_i(\vec{x},y)$.

(iii) $\frac{\partial}{\partial p(y|\vec{x})} \lambda_{m+1} \left(\sum_{y\in\mathcal{Y}} p(y|\vec{x}) - 1\right) = \lambda_{m+1}$

From Naive Bayes, HMM, ME to CRF IV

ME !

Equating the sum of the above terms to zero and solving by $p(y|x)$ leads to:

$$\begin{aligned} 0 &= -\tilde{p}(\vec{x})(\log p(y|\vec{x})+1)+\sum_{i=1}^{m}\lambda_i\tilde{p}(\vec{x})f_i(\vec{x},y)+\lambda_{m+1} \\ \log p(y|\vec{x})+1 &= \sum_{i=1}^{m}\lambda_i f_i(\vec{x},y)+\frac{\lambda_{m+1}}{\tilde{p}(\vec{x})} \\ p(y|\vec{x}) &= \exp\left(\sum_{i=1}^{m}\lambda_i f_i(\vec{x},y)\right)\cdot\exp\left(\frac{\lambda_{m+1}}{\tilde{p}(\vec{x})}-1\right). \end{aligned}$$

Subsequently, using the constraint $\sum_{y\in\mathcal{Y}} p(y|\vec{x})=1$ leads to
$\sum_{y\in\mathcal{Y}}\exp\left(\sum_{i=1}^{m}\lambda_i f_i(\vec{x},y)\right)\cdot\exp\left(\frac{\lambda_{m+1}}{\tilde{p}(\vec{x})}-1\right)=1$, and derives the result in (7).

From Naive Bayes, HMM, ME to CRF V

ME !

Generalized Iterative Scaling (GIS) for computing λ_i in (7).

定理 (GIS algorithm)

Assume $C=\max\sum_{i=1}^{m}f_i(\vec{x},y)$,

$$\lambda_i^{(t+1)}=\lambda_i^{(t)}+\frac{1}{C}\log\frac{E(f_i)}{\tilde{E}(f_i)},$$

until $\tilde{H}(Y|X)=-\sum_{\vec{x},y\in\mathcal{X}_{train}\times\mathcal{Y}}\tilde{p}(\vec{x},y)\log p^*(y|\vec{x})$ *convergences.*

A. L. Berger. The improved iterative scaling algorithm: A gentle introduction, Technical report, Carnegie Mellon University, 1997

From Naive Bayes, HMM, ME to CRF VI

ME !

Note 1: There are two core ideas in ME. First, INFERENCE !
In another word, to obtain $p^*(y|\vec{x})$ for the sake of inference;
Second, TRAINING !
Say, to obtain $\vec{\lambda}$ during iterative training steps.

Note 2: Recall the potential equation $\Psi_C(\boldsymbol{x}_C)$ defined in equation (2). In the above case, $\boldsymbol{x} = \{x, y\}$, and the only clique in $\boldsymbol{x}$ is itself. So the results in equation (7) suggests that exponential of feature functions

$$\exp\left(\sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}_C)\right)$$

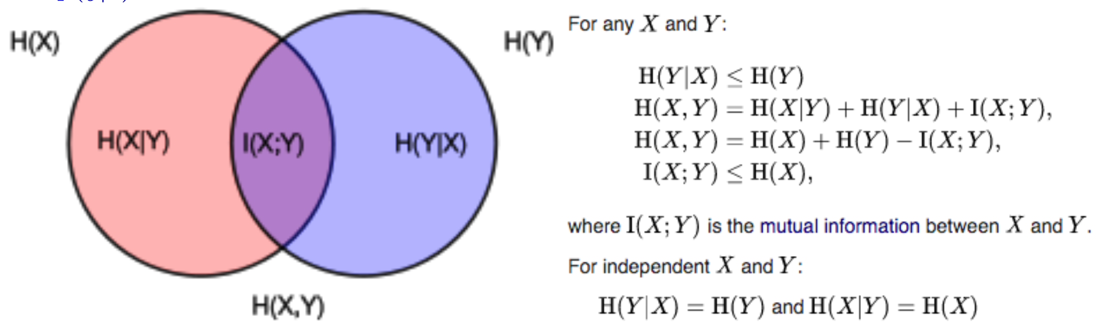
is actually a gorgeous candidate for potential function.

From Naive Bayes, HMM, ME to CRF VII

ME !

Check the below footnotes for more.

[9]Check https://en.wikipedia.org/wiki/Conditional_entropy to know more about conditional entropy. Maximize the conditional entropy is to approximate the randomness of the 'system' $p(y|x)$.



For any X and Y:

$$\begin{aligned} &\mathrm{H}(Y|X) \le \mathrm{H}(Y)\\ &\mathrm{H}(X,Y) = \mathrm{H}(X|Y) + \mathrm{H}(Y|X) + \mathrm{I}(X;Y),\\ &\mathrm{H}(X,Y) = \mathrm{H}(X) + \mathrm{H}(Y) - \mathrm{I}(X;Y),\\ &\mathrm{I}(X;Y) \le \mathrm{H}(X), \end{aligned}$$

where $\mathrm{I}(X;Y)$ is the mutual information between X and Y.

For independent X and Y:

$\mathrm{H}(Y|X) = \mathrm{H}(Y)$ and $\mathrm{H}(X|Y) = \mathrm{H}(X)$

The area contained by both circles is the joint entropy $H(X, Y)$. The circle on the left is the individual entropy $H(X)$, with the red being the conditional entropy $H(X|Y)$. The circle on the right is $H(Y)$, with the blue being $H(Y|X)$. The violet is the mutual information $I(X;Y)$.

[10]$H(Y|X) = 0$ iff. the value of Y is completely determined by the value of X, i.e., $y = g(x)$. https://math.stackexchange.com/questions/3383199/conditional-entropy-if-hyx-0-then-there-exists-a-function-g-such-that

[11]Check https://en.wikipedia.org/wiki/Mutual_information to know more about mutual information $I(X;Y) = \mathbb{KL}(p(X,Y)||p_X p_Y)$.

From Naive Bayes, HMM, ME to CRF

Coding tutorial: ME on text classification

Tutorial of ME on text classification: `https://github.com/bionlp-hzau/MaximumEntroy4Classification`.

The above coding helps to understand

1. The design of $f_i(\vec{x}, y)$ in a text classification task;
2. Compute $E(f_i)$ and $\tilde{E}(f_i)$;
3. Compute $p(y|x)$ with given $\vec{x}$ and y, (Check formula (7));
4. Iteration of λ_i with the GIS algorithm.

From Naive Bayes, HMM, ME to CRF

CRF

Finally, we are ready to introduce CRF.
First, introduce the general form of CRF.
Followed with linear chain CRF.
And the clue to read the rest parts. Enjoy!

From Naive Bayes, HMM, ME to CRF

CRF: General form of p(y|x) for an un-directional graphical model

In the view of an un-directional graphical model, we model the joint probability $p(\vec{x}, \vec{y})$ by using potential function (2), i.e., $\Psi_C(\vec{x}_C, \vec{y}')$, and we have:

$$p(\vec{y}|\vec{x}) = \frac{p(\vec{x}, \vec{y})}{p(\vec{x})} = \frac{p(\vec{x}, \vec{y})}{\sum_{\vec{y}'} p(\vec{y}', \vec{x})} = \frac{\frac{1}{Z} \prod_{C \in \mathcal{C}} \Psi_C(\vec{x}_C, \vec{y}_C)}{\frac{1}{Z} \sum_{\vec{y}'} \prod_{C \in \mathcal{C}} \Psi_C\left(\vec{x}_C, \vec{y}'_C\right)},$$

From this, the general model formulation is derived:

$$p(\vec{y}|\vec{x}) = \frac{1}{Z(\vec{x})} \prod_{C \in \mathcal{C}} \Psi_C(\vec{x}_C, \vec{y}_C), \tag{8}$$

where $Z(\vec{x}) = \sum_{\vec{y}'} \prod_{C \in \mathcal{C}} \Psi_C(\vec{x}_C, \vec{y}')$, C is a clique.

From HMM to CRF, in the View of Graphical Model I

CRF: Typical form of ME, as an example

In the ME model, $\{\vec{x}, y\}$ is the only clique in the graph.
So we have

$$p^*_{\vec{\lambda}}(y|\vec{x}) = \frac{1}{Z_{\vec{\lambda}}(\vec{x})} \Psi_C\left(\vec{x}_C, \vec{y}'\right) = \frac{1}{Z_{\vec{\lambda}}(\vec{x})} \exp\left(\sum_{i=1}^{m} \lambda_i f_i(\vec{x}, y)\right),$$

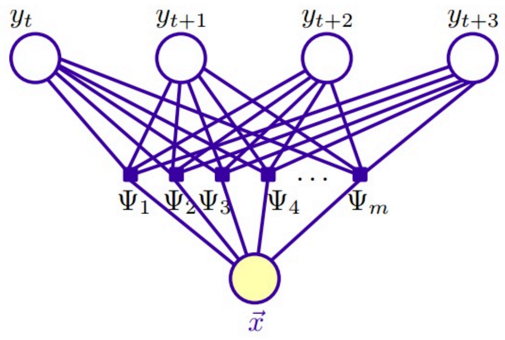
where $Z_{\vec{\lambda}}(\vec{x}) = \sum_{y \in \mathcal{Y}} \exp\left(\sum_{i=1}^{m} \lambda_i f_i(\vec{x}, y)\right)$.
Please compare this with formula (7).

As we mentioned before, ME contributes a main idea of modeling potential function with respect to $f_i(x, y)$.

From HMM to CRF, in the View of Graphical Model I

CRF: Typical form of linear chain CRF

In a linear chain CRF, all cliques are the same form, $\{y_{j-1}, y_j, \vec{x}\}$.



Based on (8), the graph factor result yields to:

$$p(\vec{y}|\vec{x}) = \frac{\prod_{j=1}^{n} \Psi_j(y_{j-1}, y_j, \vec{x})}{Z_\lambda(\vec{x})}, \tag{9}$$

From HMM to CRF, in the View of Graphical Model II

CRF: Typical form of linear chain CRF

where the factor $\Psi_j(y_{j-1}, y_j, \vec{x})$ is very alike in (7) with ME model[12]:

$$\Psi_j(y_{j-1}, y_j, \vec{x}) = \exp\left(\sum_{i=1}^{m} \lambda_i f_i\left(y_{j-1}, y_j, \vec{x}, j\right)\right), \tag{10}$$

$$\text{and, } Z_\lambda(\vec{x}) = \sum_{\vec{y}' \in \mathcal{Y}} \exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_i f_i\left(y'_{j-1}, y'_j, \vec{x}, j\right)\right). \tag{11}$$

[12] Actually, this form makes sure that it follows an exponential family distribution, where $p(x) = h(x)\exp(T(x)'\eta - A(\eta))$, η is called as natural parameter, $T(x)$ is called as sufficient statistics, $A(\eta)$ is log-normalizer, $h(x)$ is base measure.
Refer to my course on *LDA and Variational Inference* and know more.

From HMM to CRF, in the View of Graphical Model

CRF: Typical form of linear chain CRF

Note: (9) is actually regarded as

$$p(\vec{y}|\vec{x}) = \frac{\exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m}\lambda_i f_i\left(y_{j-1}, y_j, \vec{x}, j\right)\right)}{\sum_{\vec{y}'\in\mathcal{Y}}\exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m}\lambda_i f_i\left(y'_{j-1}, y'_j, \vec{x}, j\right)\right)} = \frac{\exp\left(\text{Score}(\vec{x}, \vec{y})\right)}{\sum_{\vec{y}'\in\mathcal{Y}}\exp\left(\text{Score}(\vec{x}, \vec{y}')\right)},$$

where

$$\text{Score}(\vec{x}, \vec{y}) = \sum_{j=1}^{n}\log\Psi_j(y_{j-1}, y_j, \vec{x}). \qquad (12)$$

Note: Literally, greater λ_i along with $f_i\left(y_{j-1}, y_j, \vec{x}, j\right)$ leads to higher "Score" of clique $\{y_{j-1}, y_j, \vec{x}\}$, and (12) reflects sortof cumulative score of $p(\vec{y}|\vec{x})$.

From HMM to CRF, in the View of Graphical Model

CRF

How to read the rest parts of this slides

Knowing formulas in the above pages is not sufficient to understand the whole CRF, though it shows a good sketch of the algorithm.
We would enunciate core details in the rest sections.

- ☐ Section 4 is for path inference (retrieving $\vec{y}$) by using Viterbi.
- ☐ Section 5 is for iterating the parameter λ_i in Ψ_i.
- ☐ Section 6 explains how CRF layer is introduced in the manner of neural network implementation.

Outline

Viterbi on path inference

What is inference for?

The problem of inference is to find the most likely sequence y for given observations x.

Since Viterbi algorithm is applied in HMM with a much easier-to-understand manner, we introduce this idea starting from HMM.
Enjoy!

Viterbi on path inference

Viterbi on HMM: Model parameters [13]

Observation sequence $X = \{x_1, x_2, \cdots, x_T\}$, denote $x_t = o_i$ as $x_t = i$;
where $t = 1, 2, \cdots, T$ refers to time; $o_i \in \mathcal{O}$, Observation space,
$\mathcal{O} = \{o_1, o_2, \cdots, o_N\}$, $(i = 1, 2, \cdots, N)$;
State sequence $Y = \{y_1, y_2, \cdots, y_T\}$,
where $y_t \in \mathcal{S}$, States space, $\mathcal{S} = \{s_1, s_2, \cdots, s_K\}$, $(k = 1, 2, \cdots, K)$;

Define Transition matrix $A \in \mathbb{R}^{K\times K}$, where $a_{k_1 k_2} = P(s_{k_2}|s_{k_1})$;
emission matrix $B \in \mathbb{R}^{K\times N}$, where $b_{ki} = P(o_i|s_k)$;
initial probabilities $\Pi = \{\pi_1, \pi_2, \cdots, \pi_K\}$, where $\pi_k = P(y_1 = s_k)$

HMM task
Input: $X, \lambda = \{\Pi, A, B\}$
Output: Y

[13] https://en.wikipedia.org/wiki/Viterbi_algorithm

Viterbi on path inference I

Viterbi on HMM: Viterbi Prob

Define Viterbi prob, which is called as $\delta_t(k)$ in a more popular cases,

$$V_{t,k} = \max_{y_{1:t-1}} P(y_1, y_2, \cdots, y_t = s_k, x_1, x_2, \cdots, x_t). \tag{13}$$

Note: (1) It is easy to find
$V_{1,k} = \max_{y_0} P(y_1 = s_k, x_1) = \max_{y_0} P(x_1|y_1 = s_k)P(y_1 = s_k)$
$= P(x_1|y_1 = s_k)P(y_1 = s_k) := b_{k1} \cdot \pi_k.$
(2) When computing $V_{2,k}$, use d-separation rule in the directional graphical model,

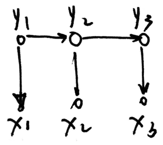
$$\begin{tikzcd} y_1 \arrow[r] \arrow[d] & y_2 \arrow[d] & \text{states} \\ x_1 & x_2 & \text{observation} \end{tikzcd}$$

we have
$V_{2,k} = \max_{y_1\in\mathcal{S}} P(y_1, y_2 = s_k, x_1, x_2) = \max_{y_1\in\mathcal{S}} P(x_1, y_1) \cdot P(y_2 = s_k|y_1)P(x_2|y_2 = s_k)$
$= \max_{y_1=s_y\in\mathcal{S}} P(x_1, y_1 = s_y) \cdot P(y_2 = s_k|y_1 = s_y)P(x_2|y_2 = s_k) = \max_{y_1=s_y\in\mathcal{S}} V_{1,y} \cdot a_{yk}b_{k2}$

Viterbi on path inference II

Viterbi on HMM: Viterbi Prob

(3) Similarly, when computing $V_{3,k}$,



we have $V_{3,k} = \max\limits_{y_1,y_2\in\mathcal{S}} P(y_1,y_2,y_3=s_k,x_1,x_2,x_3)$

$= \max\limits_{y_2=s_y\in\mathcal{S}} P(y_1,y_2=s_y,x_1,x_2)\cdot P(y_3=s_k|y_2=s_y)P(x_3|y_3=s_k)$

$= \max\limits_{y_2=s_x\in\mathcal{S}} V_{2,y}\cdot a_{yk}\cdot b_{k3}$.

(4) In all, we obtain the recursive formula to form the inductive step:

$$V_{t,k} = \max_{y_{t-1}=s_y\in\mathcal{S}} V_{t-1,y}\cdot a_{yk}\cdot b_{kt}, \qquad (14)$$

and obtain $V_{t,k}$ for $t=1,2,\cdots,T$ in a sequential order, for any given k.

Viterbi on path inference I

Viterbi on HMM: Path Retrieval

Retrieve the path Y:

Step 1: $y_T = \arg\max\limits_{s_y\in\mathcal{S}} V_{T,y}$;

Step 2: Assume $y_T=s_k$, so $V_{T,k} = \max\limits_{y_1,\cdots,y_{T-1}\in\mathcal{S}} P(y_1,\cdots,y_T=s_k,x_1,\cdots,x_T)$

$= \cdots \text{equation } (14) \cdots = \max\limits_{y_{T-1}=s_y\in\mathcal{S}} V_{T-1,y}\cdot a_{yk}\cdot b_{kT}$.

So, $y_{T-1} = \arg\max\limits_{s_y\in\mathcal{S}} V_{T-1,y}\cdot a_{yk}$.

Step 3: Again, we assume $y_{T-1}=s_k$, and compute $y_{T-2} = \arg\max\limits_{s_y\in\mathcal{S}} V_{T-2,y}\cdot a_{yk}$.

Recursively, we obtain the path Y.

The pseudocode of HMM is rewritten in the next page.

Viterbi on path inference II

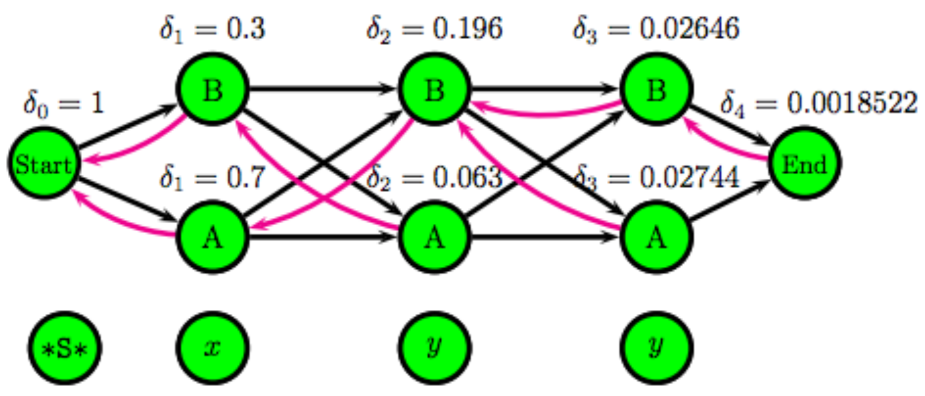
Viterbi on HMM: Path Retrieval

```
function VITERBI $(O, S, \Pi, X, A, B) : Y$
    for each state $j \in \{1, 2, \ldots, K\}$ do
        $T_1[j, 1] \leftarrow \pi_j \cdot B_{jx_1}, \; T_2[j, 1] \leftarrow 0$
    end for
    for each observation $i = 2, 3, \ldots, T$ do
        for each state $j \in \{1, 2, \ldots, K\}$ do
            $T_1[j, i] \leftarrow \max_k (T_1[k, i-1] \cdot A_{kj} \cdot B_{jx_i})$
            $T_2[j, i] \leftarrow \arg\max_k (T_1[k, i-1] \cdot A_{kj})$
        end for
    end for
    $z_T \leftarrow \arg\max_k (T_1[k, T])$
    $y_T \leftarrow s_{z_T}$
    for $i \leftarrow T, T-1, \cdots 2$ do
        $z_{i-1} \leftarrow T_2[z_i, i], \; y_{i-1} \leftarrow s_{z_{i-1}}$
    end for
    return $Y$
end function
```

Viterbi on path inference

Viterbi on HMM: Path Retrieval

Toy example



http://idiom.ucsd.edu/~rlevy/teaching/winter2009/ligncse256/lectures/hmm_viterbi_mini_example.pdf

Viterbi on path inference
Viterbi on HMM: Wrap up

Note that Viterbi is the main algorithm for tracing the path, by defining in equation (13) (Often called as $\delta_t(k)$):

$$V_{t,k} = \max_{y_{1:t-1}} P(y_1, y_2, \cdots, y_t = s_k, x_1, x_2, \cdots, x_t)$$

Besides that, an alternative algorithm, "Forward-backward" algorithm, defines a similar forward variable $\alpha_t(k)$,

$$\alpha_t(k) = \sum_{y_{1:t-1}} P(y_1, y_2, \cdots, y_t = s_k, x_1, x_2, \cdots, x_t). \tag{15}$$

Apparently, the definition of forward variable and Viterbi prob differ in summing and max.

I skip "forward-backward" algorithm for the sake of simplicity. But we will discuss it when doing CRF parameter estimation. Check page 54.

Viterbi on path inference
Viterbi on CRF

Though equation (9) offers a way to compute the discriminative probability, the numerator $Z_\lambda(\vec{x}) = \sum_{\vec{y'} \in \mathcal{Y}} \exp\left(\sum_{j=1}^{n} \sum_{i=1}^{m} \lambda_i f_i\left(y'_{j-1}, y'_j, \vec{x}, j\right)\right)$ is far from easy computation. So we need a fast implementation way.
Viterbi on CRF is pretty close to the idea of Viterbi on HMM.
The quantity $\delta_j(s|\vec{x})$, which is the highest score along a path, at position j, which ends in state s, is defined as

$$\delta_j(s|\vec{x}) = \max_{y_1, y_2, \ldots, y_{j-1}} p\left(y_1, y_2, \ldots, y_j = s|\vec{x}\right)$$

From CRF factor (see (9)), we obtain the induction step:

$$\delta_{j+1}(s|\vec{x}) = \max_{s' \in S} \delta_j\left(s'\right) \cdot \Psi_{j+1}\left(\vec{x}, s', s\right)$$

The array $\psi_j(s)$ keeps track of the best state of y_{j-1} which make $y_j = s$. The algorithm then works as follows:

Viterbi on path inference I

Viterbi on CRF: Algorithm

Step 1 Initialization:
The values for all steps from the start state $\perp$ to all possible first states s are set to the corresponding factor value

$$\begin{aligned}\forall s \in S : \delta_1(s) &= \Psi_1(\perp, s, \vec{x}) \\ \psi_1(s) &= \perp\end{aligned}$$

Step 2 Recursion:
The values for the next steps are computed from the current value and the maximum values regarding all possible succeeding states s .

$$\begin{aligned}\forall s \in S : 1 \leq j \leq n : \delta_j(s) &= \max_{s' \in S} \delta_{j-1}(s')\, \Psi_j(s', s, \vec{x}) \\ \psi_j(s) &= \operatorname*{argmax}_{s' \in S} \delta_{j-1}(s')\, \Psi_j(s', s, \vec{x})\end{aligned}$$

Viterbi on path inference II

Viterbi on CRF: Algorithm

Step 3 Termination:

$$\begin{aligned}p^* &= \max_{s' \in S} \delta_n(s') \\ y_n^* &= \operatorname*{argmax}_{s' \in S} \delta_n(s')\end{aligned}$$

Step 4 Path (state sequence) backtracking:
Recompute the optimal path from the lattice using the track keeping values ψ_t.

$$y_j^* = \psi_{j+1}\left(y_{j+1}^*\right) \quad j = n-1, n-2, \ldots, 1$$

让我们给一个人名标注的例子：https://cloud.tencent.com/developer/article/1545851。

Outline

Forward-backward Algorithm on CRF, for Training

What is training for?

Besides inference problem, the implementation of CRF faces another problem, estimation of λ_i in $\Psi_j(\vec{x}, \vec{y})$ during training, see formula (10) [14]. So, training is to obtain $\vec{\lambda}$

[14] This sub-section is fully cited again from Roman and Katrin 2007. Please check the paper for missing computation steps.

Forward-backward Algorithm on CRF, for Training

For all types of CRFs, as well as for Maximum Entropy Models, the maximum-likelihood method can be applied for parameter estimation. So, training the model is done by maximizing the log-likelihood $\mathcal{L}$ on the training data $\mathcal{T}$:

$$\begin{aligned}\overline{\mathcal{L}}(\mathcal{T}) &= \sum_{(\vec{x},\vec{y})\in\mathcal{T}} \log p(\vec{y}|\vec{x}) \\ &= \sum_{(\vec{x},\vec{y})\in\mathcal{T}} \left[\log\left(\frac{\exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m}\lambda_i f_i\left(y_{j-1},y_j,\vec{x},j\right)\right)}{\sum_{\vec{y}'\in\mathcal{Y}}\exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m}\lambda_i f_i\left(y'_{j-1},y'_j,\vec{x},j\right)\right)}\right)\right].\end{aligned}$$

and by adding the regulizer to avoid over fitting, we have

$$\mathcal{L}(\mathcal{T}) = \overline{\mathcal{L}}(\mathcal{T}) - \sum_{i=1}^{m}\frac{\lambda_i^2}{2\sigma^2}.$$

Forward-backward Algorithm on CRF, for Training I

Solve the derivative of $\mathcal{L}(\mathcal{T})$ led to

定理 (Derivative of loss function with respect to lambda)

$$\begin{aligned}\frac{\partial\mathcal{L}(\mathcal{T})}{\partial\lambda_k} &= \underbrace{\sum_{(\vec{x},\vec{y})\in\mathcal{T}}\sum_{j=1}^{n} f_k\left(y_{j-1},y_j,\vec{x},j\right)}_{\tilde{E}(f_k)} \\ &\quad - \underbrace{\sum_{(\vec{x},\vec{y})\in\mathcal{T}}\sum_{\vec{y}'\in\mathcal{Y}} p_{\vec{\lambda}}\left(\vec{y}'|\vec{x}\right)\sum_{j=1}^{n} f_k\left(y'_{j-1},y'_j,\vec{x},j\right)}_{E(f_k)} \\ &\quad - \frac{\lambda_k}{\sigma^2}\end{aligned}$$

So, λ could be computed via gradient descent:

$$\lambda^{(t+1)} = \lambda^{(t)} + \eta\nabla_\lambda(\mathcal{L}(\mathcal{T})).$$

Forward-backward Algorithm on CRF, for Training II

Now it is time to compute the derivative of the loss function.
First, $\tilde{E}(f_i)$ is the empirical distribution of a feature f_i, which is easy to obtain through direct counting in $\mathcal{T}$.
Second, "forward backward" algorithm is used to compute expectation of f_i, $E(f_i)$

Similar as the notation in equation (15), we define forward variable $\alpha_j(s|\vec{x})$ and backward variable $\beta_j(s|\vec{x})$ from $P(y_j = s|\vec{x})$:

Forward-backward Algorithm on CRF, for Training III

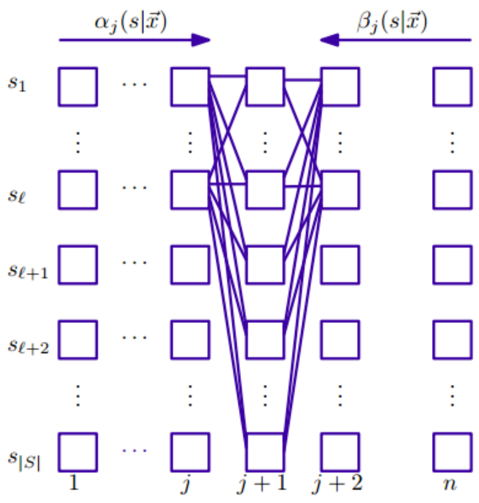


图 6: Message passing in the Forward-Backward Algorithm. Each column represents one variable, each box in a row one possible value of that variable.

Forward-backward Algorithm on CRF, for Training IV

$$
\begin{aligned}
P(y_j = s|\vec{x}) &= \sum_{y_{\neq j}\in\mathcal{Y}} P(y_{\neq j}, y_j = s|\vec{x}) \\
&= \sum_{y_{1:j-1}\in\mathcal{Y}} \sum_{y_{j+1:n}\in\mathcal{Y}} \frac{1}{Z} \prod_{j'=1}^{n} \Psi_{j'}(y_{j'-1}, y_{j'}, \vec{x}) \qquad \text{(From equation (8).)} \\
&= \frac{1}{Z}\cdot \underbrace{\sum_{y_{1:j-1}\in\mathcal{Y}} \Psi_1(y_0, y_1, \vec{x})\cdots\Psi_{j-1}(y_{j-2}, y_{j-1}, \vec{x})\Psi_j(y_{j-1}, y_j = s, \vec{x})}_{\alpha_j(s|\vec{x})} \\
&\quad \cdot \underbrace{\sum_{y_{j+1:n}\in\mathcal{Y}} \Psi_{j+1}(y_j = s, y_{j+1}, \vec{x})\Psi_{j+2}(y_{j+1}, y_{j+2}, \vec{x})\cdots\Psi_n(y_{n-1}, y_n, \vec{x})}_{\beta_{j+1}(s|\vec{x})} \qquad (16) \\
&= \frac{1}{Z}\cdot\sum_{y_{j-1}} \Psi_j(y_{j-1}, y_j = s, \vec{x})(\sum_{y_{j-2}} \Psi_{j-1}(y_{j-2}, y_{j-1}, \vec{x})\cdot\cdot(\sum_{y_0}\Psi_1(y_0, y_1, \vec{x})\cdot)) \\
&\quad \cdot\sum_{y_{j+1}} \Psi_{j+1}(y_j = s, y_{j+1}, \vec{x}) \sum_{y_{j+2}} \Psi_{j+2}(y_{j+1}, y_{j+2}, \vec{x})\cdots\sum_{y_n}\Psi_n(y_{n-1}, y_n, \vec{x}) \\
&:= \frac{1}{Z}\cdot\alpha_j(s|\vec{x})\cdot\beta_{j+1}(s|\vec{x})
\end{aligned}
$$

Forward-backward Algorithm on CRF, for Training V

we then have:

定理 (The iterative computation for forward and backward variables)

$$
\begin{aligned}
\alpha_j(s|\vec{x}) &= \sum_{s'\in T_j^{-1}(s)} \alpha_{j-1}(s'|\vec{x})\cdot\Psi_j(y_{j-1} = s', y_j = s, \vec{x}) \\
\beta_j(s|\vec{x}) &= \sum_{s'\in T_j(s)} \beta_{j+1}(s'|\vec{x})\cdot\Psi_j(y_{j-1} = s, y_j = s', \vec{x})
\end{aligned},
$$

where $T_j(s)$ maps a single state s at an input position j to a set of allowed next states at position $j+1$, and the inverse function $T_j^{(-1)}(s)$ maps the set of all states of possible predecessors to s.

The α functions are messages sent from the beginning of the chain to the end. The β functions are messages sent from the end of the chain to the beginning. They are initialized by

$$
\begin{aligned}
\alpha_0(\perp|\vec{x}) &= 1 \\
\beta_{|\vec{x}|+1}(\top|\vec{x}) &= 1
\end{aligned}.
$$

Forward-backward Algorithm on CRF, for Training VI

Results in (16) and similar computation straightforwardly yields to:

定理

$$P(y_j = s|\vec{x}) = \frac{1}{z} \cdot \alpha_j(s|\vec{x}) \cdot \beta_{j+1}(s|\vec{x}),$$

$$P(y_{j-1} = s, y_j = s'|\vec{x}) = \frac{1}{z} \cdot \alpha_{j-1}(s|\vec{x}) \cdot \Psi_j(y_{j-1} = s, y_j = s', \vec{x}) \cdot \beta_{j+1}(s'|\vec{x}).$$

With these messages, it is possible to compute the expectation $E(f_i) = \sum_{(\vec{x},\vec{y})\in\mathcal{T}} \sum_{\vec{y}'\in\mathcal{Y}} p_{\vec{\lambda}}(\vec{y}'|\vec{x}) \sum_{j=1}^{n} f_i(y'_{j-1}, y'_j, \vec{x}, j)$ under the model distribution efficiently via

$$E(f_i) = \sum_{(\vec{x},\vec{y})\in\mathcal{T}} \frac{1}{Z_{\vec{\lambda}}(\vec{x})} \sum_{j=1}^{n} \sum_{s\in S} \sum_{s'\in T_j(s)} f_i(s, s', \vec{x}, j) \cdot \alpha_{j-1}(s|\vec{x}) \Psi_j(s, s', \vec{x}) \beta_{j+1}(s'|\vec{x})$$

Forward-backward Algorithm on CRF, for Training VII

The normalization factor is computed by

$$Z_{\vec{\lambda}}(\vec{x}) = \beta_0(\perp|\vec{x})$$

Outline

CRF and Sequence Labelling—Implementation Topics I

Wapiti: A simple and fast discriminative sequence labelling toolkit

Wapiti (https://wapiti.limsi.fr/) is a very fast toolkit for segmenting and labeling sequences with discriminative models. It is based on ME, maximum entropy Markov models and linear-chain CRF and proposes various optimization and regularization methods to improve both the computational complexity and the prediction performance of standard models.

CRF and Sequence Labelling—Implementation Topics II

Wapiti: A simple and fast discriminative sequence labelling toolkit

Check https://github.com/bionlp-hzau/Tutorial_4_CRF to get the tutorial of WAPITI.

The advantage of using WAPITI is to understand **how feature engineering works in CRF**.

To generate feature functions $f_i(y_{j-1}, y_j, \vec{x}, j)$ in a batch, a patter file "*.pat" is needed. Read https://wapiti.limsi.fr/manual.html#patterns, section <Patterns>.

For example, if your data is

a1	b1	c1
a2	b2	c2
a3	b3	c3

The pattern "u:%x[-1,0]/%x[+1,2]" applied at position 2 in the sequence will produce the observation "u:a1/c3".

CRF and Sequence Labelling—Implementation Topics III

Wapiti: A simple and fast discriminative sequence labelling toolkit

For better understanding, let's show an example. Assume
武汉解封了
B I O O O
is a training sample for the task of "Location tagging", then the pattern

$$U01 : \%x[0,0]$$

generate 15 feature functions:
func1 = if (output = B and feature=U01:" 武") return 1 else return 0
func2 = if (output = I and feature=U01:" 武") return 1 else return 0
func3 = if (output = O and feature=U01:" 武") return 1 else return 0
func4 = if (output = B and feature=U01:" 汉") return 1 else return 0
func5 = if (output = I and feature=U01:" 汉") return 1 else return 0
...
func14 = if (output = I and feature=U01:" 了") return 1 else return 0
func15 = if (output = O and feature=U01:" 了") return 1 else return 0

CRF and Sequence Labelling—Implementation Topics IV

Wapiti: A simple and fast discriminative sequence labelling toolkit

Some parameter setting for the binary WAPITI file[15].

- Synopsis.
  wapiti mode [options] [input] [output]
- Train mode.
  - -a | –algo <string> Select the algorithm used to train the model, specify "list" for a list of available algorithms. The first algorithm in this list is used as default.
  - -t | –nthread <integer> Set the number of thread to use, this can drastically improve performance but is very algorithm dependent. Best value is to set it to the number of core you have. Default is 1.
  - -i | –maxiter <integer> Defines the maximum number of iterations done by the training algorithm. A value of 0 means unlimited and training will continue until another stopping criterion is reached. The default is unlimited and algorithm will stop using the others criteria.
  - -p | –pattern <file> Specify the file containing the patterns for extracting features. The format of the pattern file is detailed below.
- Label mode.

CRF and Sequence Labelling—Implementation Topics V

Wapiti: A simple and fast discriminative sequence labelling toolkit

- -c | –check Assume the data to be labeled are already labeled so during the labeling process we can check our own result displaying the error rates. This doesn't affect the labeling process and output data will remain exactly the same. However, progress will be more verbose and informative: at the end of the process, for each labels, the precision, recall, and f-measure will be displayed. If you ask for N-best output, statistics are computed only on the best sequence.
- -m | –model <file> Specifies a model file to load and to use for labeling. This switch is mandatory.

[15] https://wapiti.limsi.fr/manual.html

CRF and Sequence Labelling—Implementation Topics

Wapiti: A simple and fast discriminative sequence labelling toolkit

```
"""
======== Training outtrain (this may take some time) ========
* INPUT
input_dir=outtrain
pattern_file='pat/Tok321dis.pat'
training_options=' -a sgd-l1 -t 3 -i 10 '
train_files='(cat outtrain/[A-S]*.tab)'
* OUTPUT
output_dir=eval/bio
model_file='eval/bio/Tok321dis-train-outtrain.mod'
"""

>wapiti train  -a sgd-l1 -t 3 -i 10  -p pat/Tok321dis.pat <(cat
     outtrain/[A-S]*.tab) eval/bio/Tok321dis-train-outtrain.mod

"""
* Summary
    nb train:    10856
    nb labels:   17
    nb blocks:   412938
    nb features: 7020235
"""
```

CRF and Sequence Labelling—Implementation Topics

Wapiti: A simple and fast discriminative sequence labelling toolkit

```
"""========= Inference outtrain ========
* INPUT
input_dir=outtrain
model_file='eval/bio/Tok321dis-train-outtrain.mod'
test_files='(cat outtrain/[T-Z]*.tab)'
* OUTPUT
output_dir=eval/bio
Predicted_file='eval/bio/Tok321dis-train-test-outtrain.tab' """
>wapiti label -c -m eval/bio/Tok321dis-train-outtrain.mod <(cat
     outtrain/[T-Z]*.tab) eval/bio/Tok321dis-train-test-outtrain.tab
""" *Summary
    Nb sequences   : 3464
    Token error    :  4.80%
    Sequence error: 24.08%
* Per label statistics
    O        Pr=0.97  Rc=0.98  F1=0.98
    B-Severity  Pr=0.70  Rc=0.61  F1=0.65
    B-AdverseReaction  Pr=0.80  Rc=0.88  F1=0.84
    E-AdverseReaction  Pr=0.73  Rc=0.78  F1=0.76
    I-AdverseReaction  Pr=0.78  Rc=0.49  F1=0.60
    B-Factor  Pr=0.71  Rc=0.69  F1=0.70
    E-Severity  Pr=0.59  Rc=0.59  F1=0.59
    I-Severity  Pr=0.47  Rc=0.33  F1=0.39
```

CRF and Sequence Labelling—Implementation Topics

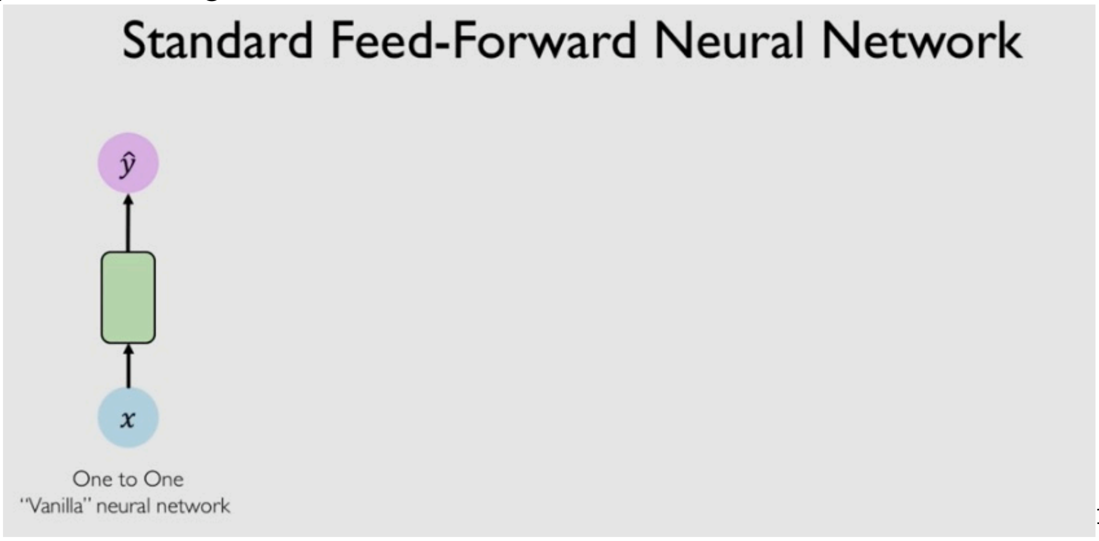
Wapiti: A simple and fast discriminative sequence labelling toolkit

```
"""
========Evaluation with conlleval.pl outtrain ========
*INPUT
Evaluate_file='eval/bio/Tok321dis-train-test-outtrain.tab'
*OUTPUT
Result_file='eval/bio/Tok321dis-train-test-outtrain.eval'
"""

>sudo perl ./conlleval.pl -d $'\t' <eval/bio/Tok321dis-train-test-
     outtrain.tab | tee eval/bio/Tok321dis-train-test-outtrain.eval

"""
*Summary
processed 60958 tok with 3669 phr; found: 3890 phr; correct: 2975.
accuracy:  95.20%; precision:  76%; recall:  81%; FB1:  78.71
  AdverseReaction: precision:  77%; recall:  84%; FB1:  80.87  3470
           Animal: precision:  76%; recall:  59%; FB1:  66.67  17
        DrugClass: precision:  18%; recall:   8%; FB1:  11.76  11
           Factor: precision:  70%; recall:  68%; FB1:  69.80  126
         Negation: precision:  57%; recall:  57%; FB1:  57.14  14
         Severity: precision:  68%; recall:  58%; FB1:  62.91  252
"""
```

CRF and Sequence Labelling—Implementation Topics I

RNN vs CRF, on sequence labelling

Traditional feed-forward network is good for classification task, but not for sequence labelling.

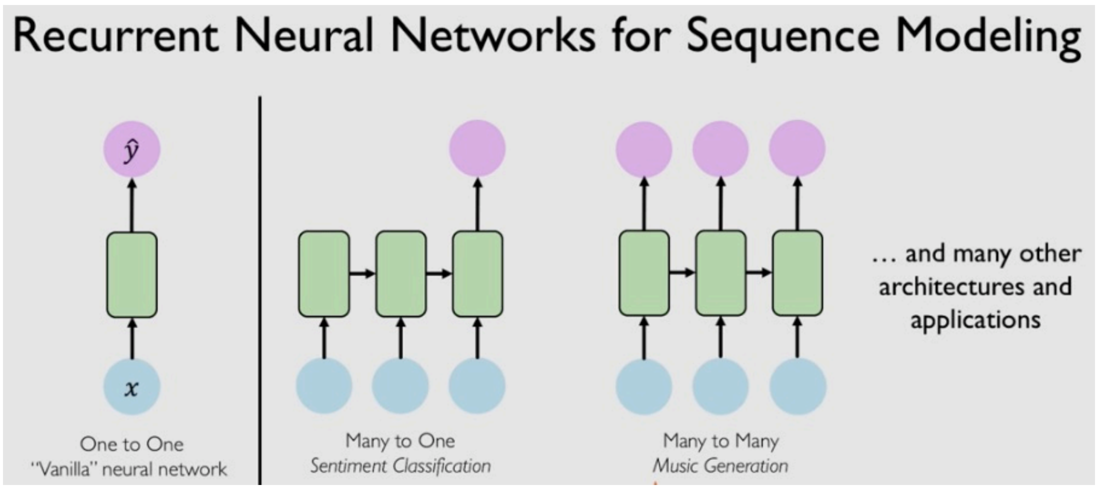


16

RNN is a suitable structure for sequence labelling by nature.

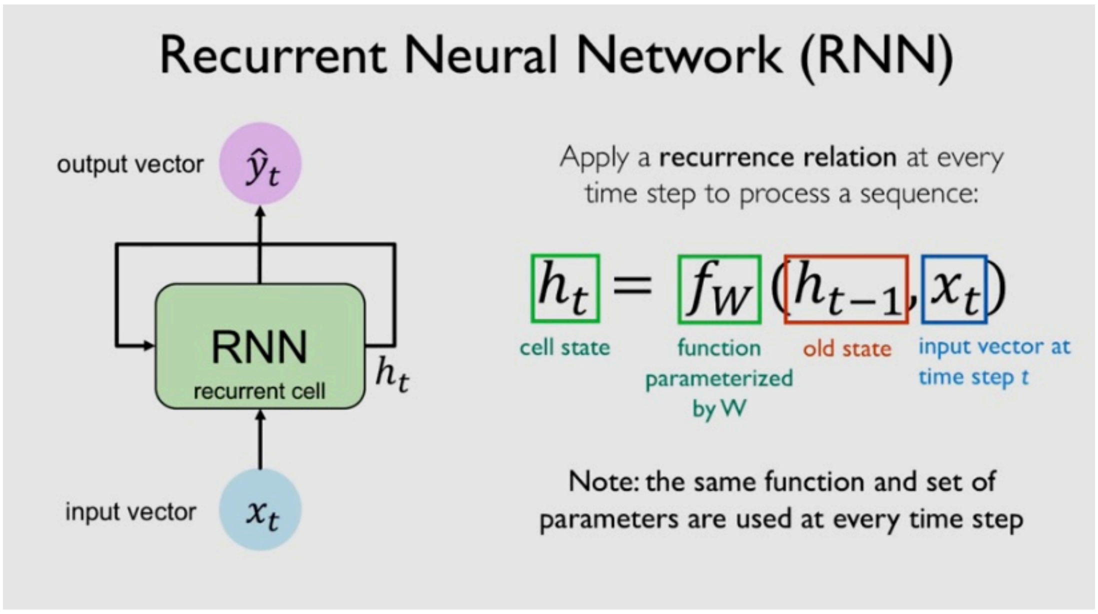
CRF and Sequence Labelling—Implementation Topics II

RNN vs CRF, on sequence labelling



CRF and Sequence Labelling—Implementation Topics III

RNN vs CRF, on sequence labelling



Apply a **recurrence relation** at every time step to process a sequence:

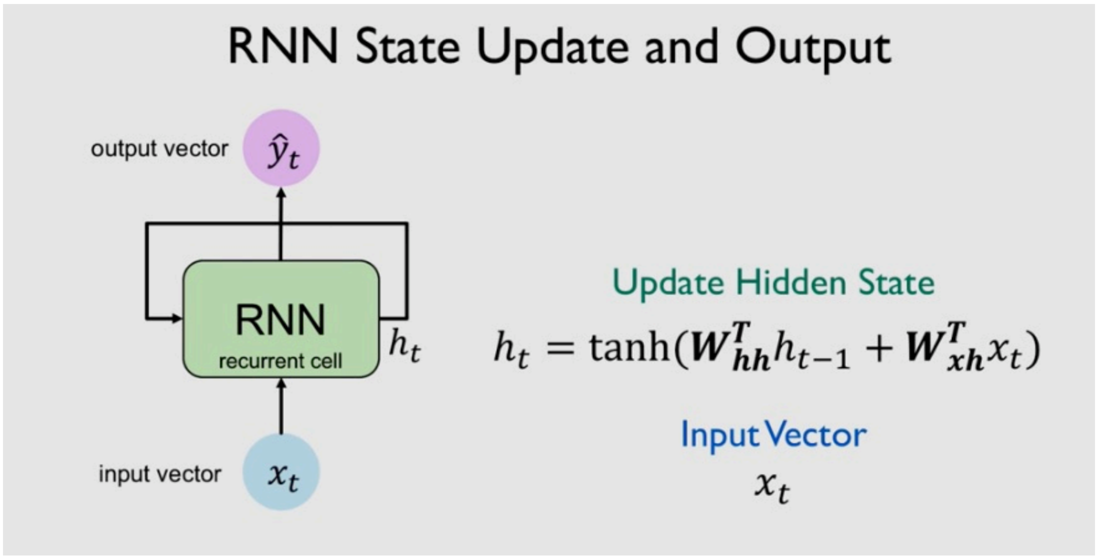
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

cell state; function parameterized by W; old state; input vector at time step t

Note: the same function and set of parameters are used at every time step

CRF and Sequence Labelling—Implementation Topics IV

RNN vs CRF, on sequence labelling



[16] http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L2.pdf

CRF and Sequence Labelling—Implementation Topics

Coding tutorial: RNN on sequence output

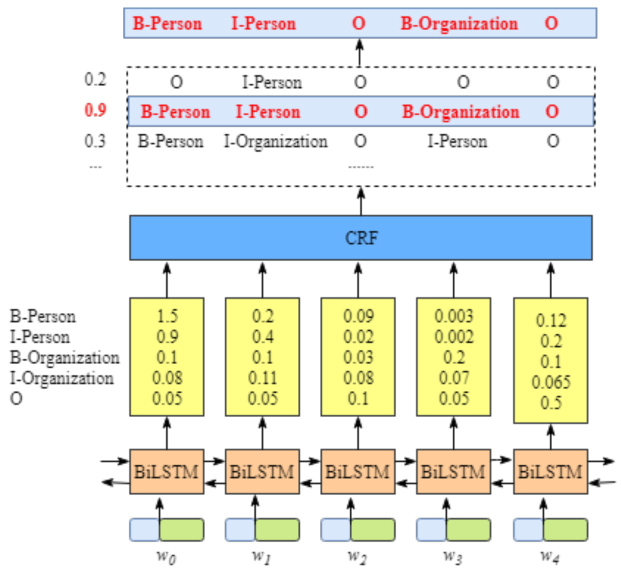
A Beginner' s Guide on Recurrent Neural Networks with PyTorch.
https://blog.floydhub.com/a-beginners-guide-on-recurrent-neural-networks-with-pytorch/

Check the above tutorial for RNN coding.
The above coding helps in understanding:

1. how a sequence input/out formalized in NLP problem.
2. one-hot coding is an extra data-processing task in NLP, but not in other scenarios like image, vision or bioinformatics, etc.
3. RNN is happy to take word embedding as original input, if they are available.

CRF and Sequence Labelling—Implementation Topics I

CRF layer in neural network, 'topping' of LSTM



CRF layer by Huang 2015 [17] learns constrains from training data, e.g.,

i) the label of the first word in a sentence starts with "B- "or "O", not "I- ";

ii) "B-Person I-Organization", "O I-label" are invalid. [18]

CRF and Sequence Labelling—Implementation Topics II

CRF layer in neural network, 'topping' of LSTM

CRF layer in NN takes the output of Bi-LSTM as the sum of weighted feature functions [19].

Recall $\text{Score}(\vec{x},\vec{y}) = \sum_{j=1}^{n} \log \Psi_j(y_{j-1}, y_j, \vec{x})$ in (12),

CRF layer defines

$$\begin{aligned}
\text{Score}(\vec{x},\vec{y}) &= \sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_i f_j(y_{j-1}, y_j, \vec{x}, j) \\
&= \sum_{j=1}^{n} \log \Psi_{EMIT}(y_j \to x_j) + \log \Psi_{TRANS}(y_{j-1} \to y_j) \\
&= \sum_{j=1}^{n} Output_{\text{Bi-LSTM}}(y_j, x_j) + Pr(y_j | y_{j-1})
\end{aligned}$$

then we have

$$p(\vec{y}|\vec{x}) = \frac{\exp(\text{Score}(\vec{x},\vec{y}))}{\sum_{\vec{y}' \in \mathcal{Y}} \exp(\text{Score}(\vec{x},\vec{y}'))} \tag{17}$$

CRF and Sequence Labelling—Implementation Topics III

CRF layer in neural network, 'topping' of LSTM

From an engineering view, there are $|\vec{x}|^{|\vec{y}|}$ possibilities for $\vec{y}$. The amount is HUGE. We need a fast implementation algorithm.

Parameter setting: Assume $\vec{x} = w_0, w_1, w_2$, Label$= l_1, l_2$.
Let's say, in a "location-tagging" task, assume $l_1 = L$, $l_2 = O$,
$\vec{x} = w_0, w_1, w_2 =$, then the expected $\vec{y} = LLO$. But there are 2^3 chances of $\vec{y}$.

The below algorithm is to compute the logarithm of numerator in (17),
$\log \sum_{\vec{y}' \in \mathcal{Y}} \exp(\text{Score}(\vec{x}, \vec{y}'))$. (Please check $TScore(w_0 \to w_1 \to w_2)$ in Step 2).)

Input: First, we have Emission matrix from output of Bi-LSTM:

$$\begin{array}{c|cc} & l_1 & l_2 \\ \hline w_0 & e_{01} & e_{02} \\ w_1 & e_{11} & e_{12} \\ w_2 & e_{21} & e_{22} \end{array}$$

, and Transition Matrix: $TRANS = \begin{array}{c|cc} & l_1 & l_2 \\ \hline l_1 & t_{11} & t_{12} \\ l_2 & t_{21} & t_{22} \end{array}$.

CRF and Sequence Labelling—Implementation Topics IV

CRF layer in neural network, 'topping' of LSTM

Step 0). w_0
$TScore(w_0) = \log(\exp(e_{01}) + \exp(e_{02}))$.

Step 1). $w_0 \to w_1$
$Obs = (e_{11}, e_{12})$
$Prev = (e_{01}, e_{02})$

*step: $\begin{pmatrix} Obs \\ Obs \end{pmatrix} \longrightarrow Obs$, $(Prev^T, Prev^T) \longrightarrow Prev$.

*step: $Score = Prev + Obs + TRANS$

$$= \begin{pmatrix} e_{01} + e_{11} + t_{11} & e_{01} + e_{12} + t_{12} \\ e_{02} + e_{11} + t_{21} & e_{02} + e_{12} + t_{22} \end{pmatrix}$$

*step: $Prev \leftarrow (\log(\exp(Score_{11}) + \exp(Score_{21})), \log(\exp(Score_{12}) + \exp(Score_{22})))$

CRF and Sequence Labelling—Implementation Topics V

CRF layer in neural network, 'topping' of LSTM

$= (\log(\exp(e_{01} + e_{11} + t_{11}) + \exp(e_{02} + e_{11} + t_{21})), \log(\exp(e_{01} + e_{12} + t_{12}) + \exp(e_{02} + e_{12} + t_{22})))$

*step: $TScore(w_0 \to w_1) = \log(\exp(Prev(1)) + \exp(Prev(2)))$

$= \log(\exp(e_{01} + e_{11} + t_{11}) + \exp(e_{02} + e_{11} + t_{21}) + \exp(e_{01} + e_{12} + t_{12}) + \exp(e_{02} + e_{12} + t_{22}))$

Step 2). $w_0 \longrightarrow w_1 \longrightarrow w_2$

$Obs = (e_{21}, e_{22})$

$Prev =$ last round

*step: $\begin{pmatrix} Obs \\ Obs \end{pmatrix} \longrightarrow Obs,\ (Prev^T, Prev^T) \longrightarrow Prev.$

*step: $Score = Prev + Obs + TRANS$

*step: $Prev \leftarrow (\log(\exp(Score_{11}) + \exp(Score_{21})), \log(\exp(Score_{12}) + \exp(Score_{22})))$

*step: $TScore(w_0 \to w_1 \to w_2) = \log(\exp(Prev(1)) + \exp(Prev(2)))$

CRF and Sequence Labelling—Implementation Topics VI

CRF layer in neural network, 'topping' of LSTM

$= \log(\exp(e_{01} + e_{11} + t_{11} + e_{21} + t_{11}) + \exp(e_{02} + e_{11} + t_{21} + e_{21} + t_{11}) + \exp(e_{01} + e_{12} + t_{12} + e_{21} + t_{21}) + \exp(e_{02} + e_{12} + t_{22} + e_{21} + t_{21}) + \exp(e_{01} + e_{11} + t_{11} + e_{22} + t_{12}) + \exp(e_{02} + e_{11} + t_{21} + e_{22} + t_{12}) + \exp(e_{01} + e_{12} + t_{12} + e_{22} + t_{22}) + \exp(e_{02} + e_{12} + t_{22} + e_{22} + t_{22}))$

Output: Obtain $\log \sum_{\vec{y'} \in \mathcal{Y}} \exp(\mathrm{Score}(\vec{x}, \vec{y'}))$ from the above term.

[17] Paper for CRF layer: https://arxiv.org/abs/1508.01991)

[18] https://createmomo.github.io/2017/09/12/CRF_Layer_on_the_Top_of_BiLSTM_1/

[19] Check "ADVANCED: MAKING DYNAMIC DECISIONS AND THE BI-LSTM CRF" https://pytorch.org/tutorials/beginner/nlp/advanced_tutorial.html or "Implementing a linear-chain Conditional Random Field (CRF) in PyTorch" https://towardsdatascience.com/implementing-a-linear-chain-conditional-random-field-crf-in-pytorch-16b0b9c4b4ea for more.

CRF and Sequence Labelling—Implementation Topics

CRF layer in neural network, 'topping' of LSTM

Check https://github.com/jidasheng/bi-lstm-crf to see how forward algorithm works in coding.

```python

    def __forward_algorithm(self, features, masks):
        """calculate the partition function with forward algorithm.
        TRICK: log_sum_exp([x1, x2, x3, x4, ...]) = log_sum_exp([
            log_sum_exp([x1, x2]), log_sum_exp([x3, x4]), ...])
        :param features: features. [B, L, C]
        :param masks: [B, L] masks
        :return:    [B], score in the log space
        """
        B, L, C = features.shape

        scores = torch.full((B, C), IMPOSSIBLE, device=features.device
            )  # [B, C]
        scores[:, self.start_idx] = 0.
        trans = self.transitions.unsqueeze(0)  # [1, C, C]
```

CRF and Sequence Labelling—Implementation Topics

CRF layer in neural network, 'topping' of LSTM

```python

        # Iterate through the sentence
        for t in range(L):
            emit_score_t = features[:, t].unsqueeze(2)  # [B, C, 1]
            score_t = scores.unsqueeze(1) + trans + emit_score_t  # [B
                , 1, C] + [1, C, C] + [B, C, 1] => [B, C, C]
            score_t = log_sum_exp(score_t)  # [B, C]
            mask_t = masks[:, t].unsqueeze(1)  # [B, 1]
            scores = score_t * mask_t + scores * (1 - mask_t)
        scores = log_sum_exp(scores + self.transitions[self.stop_idx])
        return scores
```

Please note that the above score_t [B, 1, C] , trans [1, C, C], and emit_score [B, C, 1] refer to $Prev$, $TRANS$, and Obs, respectively.

When three PyTorch tensors are added together, they are performing an addition with an extension manner, say: " [B, 1, C] + [1, C, C] + [B, C, 1] => [B, C, C]". Thus, $\begin{pmatrix} Obs \\ Obs \end{pmatrix} \longrightarrow Obs$ and $(Prev^T, Prev^T) \longrightarrow Prev$ are both implemented in an efficient way.

CRF and Sequence Labelling—Implementation Topics

CRF layer in neural network, 'topping' of LSTM

Check PyTorch official tutorial [20] for the same scripts.

```
def _forward_alg(self, feats):

        ...
        next_tag_var = forward_var + trans_score + emit_score
        ...
```

See? This is a trick for PyTorch tensor computation!

[20] https://pytorch.org/tutorials/beginner/nlp/advanced_tutorial.html

CRF and Sequence Labelling—Implementation Topics

CRF 实现和代码能力提高

- 基于 Wapiti，理解" 特征工程-训练寻参-预测" 的研究范式
  - Wapiti 是一个二进制可执行文件，请将其运行成功；
  - 尝试修改 pat 文件，理解特征函数的设置；
  - 学会划分数据集为 Train, Development 和 Test，完成特征函数寻优。
- LSTM+CRF
  - 阅读 PyTorch 官方 LSTM+CRF 的代码；
  - 导入自己的训练数据进行参数训练。
- 比较 Wapiti 和 LSTM+CRF 的性能。

Acknowledgement

Thank Pierre Zweigenbaum for the mini-course 《基于有监督学习的人工智能医药文本处理》, Oct, 2017.
Thank audiences in my graduate course, "BioTM and Knowledge Discovery, Spring 2019", https://hzaubionlp.com/course-bionlp-and-kd/, and in my undergraduate course, "NLP" Fall 2020, https://hzaubionlp.com/nlp4underg/.
Thank all HZAU BioNLP lab members who participate the algorithm discussions.

2020 年 11 月 18 日