# Conditional Random Fields
# 2. HMMs and Linear-Chain CRFs

Chunpai Wang

May 5, 2019

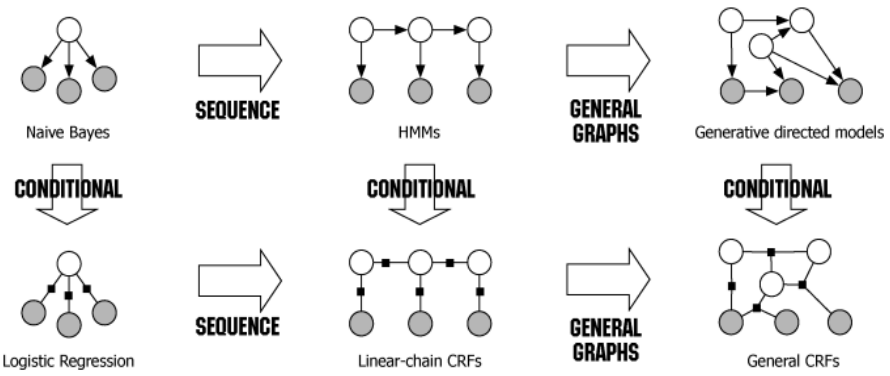


Figure 1: Diagram of the relationship between naive Bayes, logistic regression, HMMs, linear-chain CRFs, generative models, and general CRFs

## 1 Hidden Markov Model

The hidden Markov model can be visualized from figure 1. The joint distribution of observed variables $\mathbf{x}$ and unobserved variables $\mathbf{y}$ can be formulated as

$$p(\mathbf{x}, \mathbf{y}) = p(y_0)p(x_0|y_0)\prod_{i=1}^{n} p(y_i|y_{i-1})p(x_i|y_i) \tag{1}$$

There are 3 probabilities we need to consider in this scenario:

- transition probability $p(y_i|y_{i-1})$
- emission probability $p(y_i|x_i)$
- initial probability $p(y_0)$

Those three probabilities can be set by domain knowledge or learned from some existing prior knowledge. However, for following two algorithms, we assume those 3 probabilities are known.

### 1.1 Forward Backward Algorithm

Forward-Backward algorithm is used to compute the probability $p(y_k|\mathbf{x})$ in HMMs, which is an example of dynamic programming algorithm, assumed transition probabilities, emission probabilities, and initial probability are known.

Forward is to compute the $p(y_k, \mathbf{x}_{1:k})$ for all $k$, and the backward is to compute the $p(\mathbf{x}_{k+1:n}|y_k)$ for all $k$. We can see that

$$p(y_k|\mathbf{x}) \propto p(y_k, \mathbf{x}) = p(y_k, \mathbf{x}_{1:k}) \cdot p(\mathbf{x}_{(k+1):n}|y_k, \mathbf{x}_{1:k}) = p(y_k, \mathbf{x}_{1:k}) \cdot p(\mathbf{x}_{(k+1):n}|y_k) \tag{2}$$

note that $\mathbf{x}_{1:k}$ is conditionally independent to $\mathbf{x}_{(k+1):n}$ given the $y_k$. Now, we focus on computing the $p(y_k, \mathbf{x}_{1:k})$ first. The trick here is the marginalization,

$$
\begin{aligned}
p(y_k, \mathbf{x}_{1:k}) &= \sum_{y_{(k-1)}=1}^{M} p(y_k, y_{k-1}, \mathbf{x}_{1:k}) \\
&= \sum_{y_{(k-1)}=1}^{M} p(x_k|y_k, y_{k-1}, \mathbf{x}_{1:(k-1)}) \cdot p(y_k|y_{k-1}, x_{1:(k-1)}) \cdot p(y_{(k-1)}, \mathbf{x}_{1:(k-1)}) \\
&= \sum_{y_{(k-1)}=1}^{M} \underbrace{p(x_k|y_k)}_{\text{emission prob.}} \cdot \underbrace{p(y_k|y_{k-1})}_{\text{transition prob.}} \cdot p(y_{(k-1)}, \mathbf{x}_{1:(k-1)})
\end{aligned}
$$

We can see a recursion in the formula, thus we can use the Dynamic Programming to compute those distributions.

The backward algorithm is very similar to the forward algorithm.

$$
\begin{aligned}
p(\mathbf{x}_{(k+1):n}|y_k) &= \sum_{y_{(k+1)}=1}^{M} p(\mathbf{x}_{(k+1):n}, y_{(k+1)}|y_k) \\
&= \sum_{y_{(k+1)}=1}^{M} p(x_{(k+1)}|y_{(k+1)}, y_k) \cdot p(y_{(k+1)}|y_k) \cdot p(\mathbf{x}_{(k+2):n}|y_{(k+1)}, y_k, x_{(k+1)}) \\
&= \sum_{y_{(k+1)}=1}^{M} \underbrace{p(x_{(k+1)}|y_{(k+1)})}_{\text{emission prob.}} \cdot \underbrace{p(y_{(k+1)}|y_k)}_{\text{transition prob.}} \cdot p(\mathbf{x}_{(k+2):n}|y_{(k+1)})
\end{aligned}
$$

We also see a recursion in the backward formula. When we use the forward-backward algorithm on HMMs, since it needs many multiplications, it usually runs into underflow, which can be fixed by "log-sum-exp" trick.

Note that, the forward-backward algorithm is similar to the sum-product algorithm in Markov random fields

.

## 1.2 Viterbi Algorithm

Viterbi algorithm is to find the most probabilistic unobserved states given a sequence of observed states,

$$
\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y}} p(\mathbf{y}, \mathbf{x})
$$

assumed transition prob., emission prob., and initial prob. are all known. This algorithm is very similar to forward-backward algorithm, which use the recursion and dynamic programming to solve the problem.

If $f(a) \geq 0 \ \forall a$ and $g(a, b) \geq 0 \ \forall a, b$, then

$$
\max_{a,b} f(a)g(a, b) = \max_{a} \left[ f(a) \max_{b} g(a, b) \right]
$$

since

$$
\max_{b} f(a)g(a, b) = f(a) \max_{b} g(a, b)
$$

Now, we define

$$
\begin{aligned}
\mu_k(y_k) &= \max_{\mathbf{y}_{1:(k-1)}} p(\mathbf{y}_{1:k}, \mathbf{x}_{1:k}) \\
&= \max_{\mathbf{y}_{1:(k-1)}} p(x_k|y_k) \cdot p(y_k|y_{(k-1)}) \cdot p(\mathbf{y}_{1:(k-1)}, \mathbf{x}_{1:(k-1)}) \\
&= \max_{y_{k-1}} p(x_k|y_k) \cdot p(y_k|y_{(k-1)}) \cdot \max_{\mathbf{y}_{1:(k-2)}} p(\mathbf{y}_{1:(k-1)}, \mathbf{x}_{1:(k-1)}) \\
&= \max_{\mathbf{y}_{1:(k-1)}} p(x_k|y_k) \cdot p(y_k|y_{(k-1)}) \cdot \mu_{k-1}(y_{k-1})
\end{aligned}
$$

and we have $\mu_1(y_1) = p(y_1, x_1) = p(y_1)p(x_1|y_1)$, and $\mu_n(y_n) = \max_{\mathbf{y}_{1:(n-1)}} p(\mathbf{x}_{1:n}, \mathbf{y}_{1:n})$. Note that, we want to maximize $\mathbf{y}_{1:(n)}$, we have to obtain the maximal $\mathbf{y}_{1:(n-1)}$ first. Hence, we need to compute the $\mu_1(y_1)$ to $\mu_n(y_n)$ iteratively.

Note that, the viterbi algorithm is similar to the max-sum algorithm in Markov random fields

.

## 2 From HMMs to Linear-Chain CRFs

HMMs are the sequence version of Naive Bayes models, while linear-chain CRFs are the sequence version of logistic regression. HMMs assume those features are independent, like Naive Bayes. However, linear-chain CRFs can model the overlapping and non-independent features, where the sum-product algorithm and max-product algorithm still apply. (Note that, conditional random fields is the conditional probability distribution given some observed random variable $X$, which satisfy the pairwise Markov property, local Markov property, or global Markov property.)

**Definition 1.** Assume $X = (X_1, X_2, ..., X_n), Y = (Y_1, Y_2, ..., Y_n)$ be linear-chained random variables. Assume the $X$ is observed, the conditional probability distribution $p(Y|X)$ forms the conditional random field, which mean

$$p(Y_i|X, Y_1, ..., Y_{i-1}, Y_{i+1}, ..., Y_n) = p(Y_i|X, Y_{i-1}, Y_{i+1}) \tag{3}$$

then we call $p(Y|X)$ as linear-chained CRF.

Consider the name entity recognition (NER) problem, let $\mathbf{x}_{1:N}$ be the observations (e.g. words in a document), and $\mathbf{y}_{1:N}$ the hidden labels (e.g. tags). A linear chain Conditional Random Field defines a conditional probability (whereas HMM defines the joint)

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_{n=1}^{N}\sum_{i=1}^{F} \lambda_i f_i(y_{n-1}, y_n, \mathbf{x}_{1:N}, n)\right) \tag{4}$$

$$= \frac{1}{Z}\prod_{n=1}^{N} \exp\left(\sum_{i=1}^{F} \lambda_i f_i(y_{n-1}, y_n, \mathbf{x}_{1:N}, n)\right) \tag{5}$$

$$= \frac{1}{Z}\prod_{n=1}^{N} \Psi_n(y_n, y_{n-1}, \mathbf{x}_n) \tag{6}$$

Since the potential function should be strictly positive, we usually define the potential functions to be exponential functions.

where we sum over $n = 1, ..., N$ word positions in the sequence. For each position, we sum over $i = 1, ..., F$ weighted features. The scalar $\lambda_i$ is the weight for feature $f_i(\cdot)$. The $\lambda_i$ are the parameters of the CRF model, and must be learned. We can see that, we can express it as a multiplication of local functions $\Psi_n$ with cliques correspond to a pair of states $y_{n-1}, y_n$ as well as the corresponding $x_n$ nodes, with

$$\Psi_n = \exp(\lambda \cdot \mathbf{f})$$

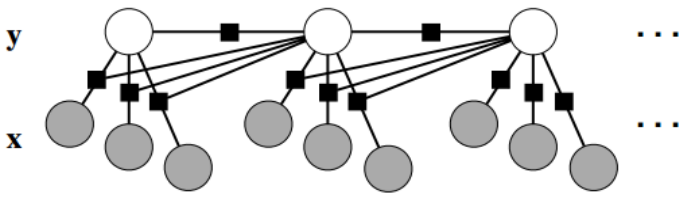


Figure 2: Graphical model of a linear-chain CRF in which the transition factors depend on the current observation. (Discussed in this section.) [1]

### 2.1 Feature Function

We use an NER example with linear-chain CRF to illustrate the feature functions, which are the key component of CRF.[2]. Note that, feature functions are not potential functions according to the equation (5). For example, we can define a simple feature function which produces binary values: it is 1 if the current word $x_n$ is John, and if the current state $y_n$ is PERSON:

$$f_1(y_{n-1}, y_n, \mathbf{x}_{1:N}, n) = \begin{cases} 1 & if\ \mathrm{y}_n = PERSON \text{ and } x_n = John \\ 0 & otherwise \end{cases} \tag{7}$$

How is this feature used ? It depends on its corresponding weight $\lambda_1$. If $\lambda_1 > 0$, whenever $f_1$ is active (i.e. we see the word John in the sentence and we assign it tag PERSON), it increases the probability of the tag sequence $\mathbf{y}_{1:N}$, which is $p(\mathbf{y}, \mathbf{x})$. Of course, $\lambda_1$ can be $\leq 0$. We may set $\lambda_1$ by domain knowledge, or learn $\lambda_1$ from corpus, or both. Note, $\lambda_1, f_1(\cdot)$ together play the same role as HMM's transition or emission probabilities. Also, we can use another feature function $f_2$ for the same word $x_n$. Furthermore, note $f_1$ and $f_2$ can be both active for a sentence. This is an example of overlapping features. It boosts up the belief of $y_1 = PERSON$ to $\lambda_1 + \lambda_2$. Of course the features are not limited to binary functions. Any real-valued function is allowed. Designing the features of an CRF is the most important task. In CRFs for real applications, it is common to have tens of thousands or more features.

Here, we define a feature function as any mapping $f_k : Y \times Y \times X \to \mathbb{R}$. As a special case, a feature function can be Boolean, that is a mapping $f_k : Y \times Y \times X \to \{0, 1\}$. We usually define classes of feature functions using templates of some sort. The integer $k$ is then defined to range over all combination of members of the classes.
We can write (18) more compactly by introducing the concept of feature functions. Each feature function has the form $f_k(y_t, y_{t-1}, x_t)$. In order to duplicate (18), we need one feature

$$f_{ij}(y, y', x) = \mathbb{1}_{\{y=i\}} \mathbb{1}_{\{y'=j\}} \tag{8}$$

for each transition $(i, j)$ and one feature

$$f_{io}(y, y', x) = \mathbb{1}_{\{y=i\}} \mathbb{1}_{\{x=o\}} \tag{9}$$

for each state-observation pair $(i, o)$. We refer to a feature function generally as $f_k$, where $f_k$ ranges over both all of the $f_{ij}$ and all of the $f_{io}$. For example, if $|Y| = 3$ and $|X| = 4$, then we have size $|f_{ij}| = 3 * 3 = 9$ and size $|f_{io}| = 3 * 4 = 12$, then $K = 9 + 12 = 21$. Then we can write an HMM as:

$$p(\mathbf{y}, \mathbf{x}) = \frac{1}{Z} \prod_{t=1}^{T} \exp \left\{ \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, x_t) \right\} \tag{10}$$

Again, this equation defines exactly the same family of distributions as (2). The last step is to write the conditional distribution $p(\mathbf{y}, \mathbf{x})$ that results from the HMM (8). This is

$$p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{y}, \mathbf{x})}{\sum_{\mathbf{y}'} p(\mathbf{y}', \mathbf{x})} = \frac{\prod_{t=1}^{T} \exp \left\{ \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, x_t) \right\}}{\sum_{\mathbf{y}'} \prod_{t=1}^{T} \exp \left\{ \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, x_t) \right\}} \tag{11}$$

Note that $Z \neq \sum_{\mathbf{y}'} \prod_{t=1}^{T} \exp \left\{ \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, x_t) \right\}$. This conditional distribution is a particular kind of linear-chain CRF, namely, one that includes features only for the current word's identity (only one $x_t$ in the input of feature function). But many other linear-chain CRFs use richer features of the input, and it only requires little change on the feature functions.

### 2.2 CRF Training

Training is to find the parameters $\lambda$ in a CRF. Please check [1] and [2] for reference.

## References


[1] Sutton, Charles, and Andrew McCallum. "An introduction to conditional random fields." Foundations and Trends® in Machine Learning 4.4 (2012): 267-373.

[2] http://pages.cs.wisc.edu/~jerryzhu/cs769/CRF.pdf