Flipped Classroom Questions on Conditional Random Fields

Michael Collins

1 Notation

Throughout this note I'll use *underline* to denote vectors. For example, $\underline{w} \in \mathbb{R}^d$ will be a vector with components $w_1, w_2, \ldots w_d$. We use $\exp(x)$ for the exponential function, i.e., $\exp(x) = e^x$.

2 CRFs

We now turn to conditional random fields (CRFs).

One brief note on notation: for convenience, we'll use $\underline{x}$ to refer to an input sequence $x_1 \ldots x_m$, and $\underline{s}$ to refer to a sequence of states $s_1 \ldots s_m$. The set of all possible states is again $\mathcal{S}$; the set of all possible state sequences is $\mathcal{S}^m$. In conditional random fields we'll again build a model of

$$p(s_1 \ldots s_m | x_1 \ldots x_m) = p(\underline{s}|\underline{x})$$

A first key idea in CRFs will be to define a feature vector

$$\underline{\Phi}(\underline{x}, \underline{s}) \in \mathbb{R}^d$$

that maps an *entire input sequence* $\underline{x}$ *paired with an entire state sequence* $\underline{s}$ to some d-dimensional feature vector. We'll soon give a concrete definition for $\underline{\Phi}$, but for now just assume that some definition exists. We will often refer to $\underline{\Phi}$ as being a "global" feature vector (it is global in the sense that it takes the entire state sequence into account).

We then build a *giant* log-linear model,

$$p(\underline{s}|\underline{x}; \underline{w}) = \frac{\exp\left(\underline{w} \cdot \underline{\Phi}(\underline{x}, \underline{s})\right)}{\sum_{\underline{s}' \in \mathcal{S}^m} \exp\left(\underline{w} \cdot \underline{\Phi}(\underline{x}, \underline{s}')\right)}$$

This is "just" another log-linear model, but it is is "giant" in the sense that: 1) the space of possible values for $\underline{s}$, i.e., $\mathcal{S}^m$, is huge. 2) The normalization constant (denominator in the above expression)

involves a sum over the set $\mathcal{S}^m$. At first glance, these issues might seem to cause severe computational problems, but we'll soon see that under appropriate assumptions we can train and decode efficiently with this type of model.

The next question is how to define $\underline{\Phi}(\underline{x}, \underline{s})$? Our answer will be

$$\underline{\Phi}(\underline{x}, \underline{s}) = \sum_{j=1}^{m} \underline{\phi}(\underline{x}, j, s_{j-1}, s_j)$$

Here $\underline{\phi}(\underline{x}, j, s_{j-1}, s_j)$ is a feature vector where:

- $\underline{x} = x_1 \dots x_m$ is the entire sentence being tagged
- j is the position to be tagged (can take any value from 1 to m)
- s_{j-1} is the previous state value (can take any value in $\mathcal{S}$)
- s_j is the new state value (can take any value in $\mathcal{S}$)

See the lecture slides on log-linear models (from Lecture 1) to see examples of features used in applications such as part-of-speech tagging.

Or put another way, we're assuming that for $k = 1 \dots d$, the k'th global feature is

$$\Phi_k(\underline{x}, \underline{s}) = \sum_{j=1}^{m} \phi_k(\underline{x}, j, s_{j-1}, s_j)$$

Thus Φ_k is calculated by summing the "local" feature vector ϕ_k over the m different state transitions in $s_1 \dots s_m$.

We now turn to two critical practical issues in CRFs: first, *decoding*, and second, *parameter estimation*.

Decoding with CRFs The decoding problem in CRFs is as follows: for a given input sequence $\underline{x} = x_1, x_2, \dots x_m$, we would like to find the most likely underlying state sequence under the model, that is,

$$\arg\max_{\underline{s} \in \mathcal{S}^m} p(\underline{s}|\underline{x}; \underline{w})$$

We simplify this expression as follows:

$$\begin{aligned} \arg\max_{\underline{s} \in \mathcal{S}^m} p(\underline{s}|\underline{x}; \underline{w}) &= \arg\max_{\underline{s} \in \mathcal{S}^m} \frac{\exp\left(\underline{w} \cdot \underline{\Phi}(\underline{x}, \underline{s})\right)}{\sum_{\underline{s}' \in \mathcal{S}^m} \exp\left(\underline{w} \cdot \underline{\Phi}(\underline{x}, \underline{s}')\right)} \\ &= \arg\max_{\underline{s} \in \mathcal{S}^m} \exp\left(\underline{w} \cdot \underline{\Phi}(\underline{x}, \underline{s})\right) \end{aligned}$$

$$
\begin{aligned}
&= \arg\max_{\underline{s} \in \mathcal{S}^m} \quad \underline{w} \cdot \underline{\Phi}(\underline{x}, \underline{s}) \\
&= \arg\max_{\underline{s} \in \mathcal{S}^m} \quad \underline{w} \cdot \sum_{j=1}^{m} \underline{\phi}(\underline{x}, j, s_{j-1}, s_j) \\
&= \arg\max_{\underline{s} \in \mathcal{S}^m} \quad \sum_{j=1}^{m} \underline{w} \cdot \underline{\phi}(\underline{x}, j, s_{j-1}, s_j)
\end{aligned}
$$

So we have shown that finding the most likely sequence under the model is equivalent to finding the sequence that maximizes

$$\arg\max_{\underline{s} \in \mathcal{S}^m} \quad \sum_{j=1}^{m} \underline{w} \cdot \underline{\phi}(\underline{x}, j, s_{j-1}, s_j)$$

Question 1: describe a dynamic programming problem that finds the $\arg\max$ in the above equation.

Parameter Estimation in CRFs. For parameter estimation, we assume we have a set of n labeled examples, $\{(\underline{x}^i, \underline{s}^i)\}_{i=1}^n$. Each $\underline{x}^i$ is an input sequence $x_1^i \dots x_m^i$, each $\underline{s}^i$ is a state sequence $s_1^i \dots s_m^i$. We then proceed in exactly the same way as for regular log-linear models. The *regularized log-likelihood function* is

$$L(\underline{w}) = \sum_{i=1}^{n} \log p(\underline{s}^i | \underline{x}^i; \underline{w}) - \frac{\lambda}{2} ||\underline{w}||^2$$

Our parameter estimates are then

$$\underline{w}^* = \arg\max_{\underline{w} \in \mathbb{R}^d} \quad \sum_{i=1}^{n} \log p(\underline{s}^i | \underline{x}^i; \underline{w}) - \frac{\lambda}{2} ||\underline{w}||^2$$

We'll again use gradient-based optimization methods to find $\underline{w}^*$. As before, the partial derivatives are

$$\frac{\partial}{\partial w_k} L(\underline{w}) \quad = \quad \sum_i \Phi_k(\underline{x}^i, \underline{s}^i) - \sum_i \sum_{\underline{s} \in \mathcal{S}^m} p(\underline{s} | \underline{x}^i; \underline{w}) \Phi_k(\underline{x}^i, \underline{s}) - \lambda w_k$$

The first term is easily computed, because

$$\sum_i \Phi_k(\underline{x}^i, \underline{s}^i) = \sum_i \sum_{j=1}^{m} \phi_k(\underline{x}^i, j, s_{j-1}^i, s_j^i)$$

Hence all we have to do is to sum over all training examples $i = 1 \dots n$, and for each example sum over all positions $j = 1 \dots m$.

The second term is more difficult to deal with, because it involves a sum over $\mathcal{S}^m$, a very large set.

Question 2: Assume that for any i, j, for any $a \in \mathcal{S}, b \in \mathcal{S}$, we can efficiently compute

$$q_j^i(a,b) = \sum_{\underline{s} \in \mathcal{S}^m : s_{j-1}=a, s_j=b} p(\underline{s}|\underline{x}^i; \underline{w})$$

The quantity $q_j^i(a,b)$ has a fairly intuitive interpretation: it is the probabilty of the i'th training example $\underline{x}^i$ having state a at position $j-1$ and state b at position j, under the distribution $p(\underline{s}|\underline{x}; \underline{w})$.

Show that given an algorithm that computes all $q_j^i(a,b)$ terms efficiently, it is possible to efficiently (in polynomial time) compute

$$\sum_i \sum_{\underline{s} \in \mathcal{S}^m} p(\underline{s}|\underline{x}^i; \underline{w}) \Phi_k(\underline{x}^i, \underline{s})$$

(Note: A critical result is that for a given i, *all* $q_j^i(a,b)$ *terms can be calculated together, in* $O(mk^2)$ *time*. The algorithm that achieves this is the *forward-backward algorithm*. This is another dynamic programming algorithm, and is closely related to the Viterbi algorithm.)