Conditional Random Fields

1 Introduction

Conditional random fields (CRFs) define a distribution over target structures $\boldsymbol{y}$ given an input structure $\boldsymbol{x}$, using a global feature function Φ and a weight vector $\boldsymbol{w}$:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{\exp(\boldsymbol{w} \cdot \Phi(\boldsymbol{x}, \boldsymbol{y}))}{\sum_{\boldsymbol{y}'} \exp(\boldsymbol{w} \cdot \Phi(\boldsymbol{x}, \boldsymbol{y}'))} \tag{1}$$

CRFs are powerful because Φ can consider an *entire* target structure $\boldsymbol{y}$. We will focus on the case where the structure is a sequence: $\boldsymbol{x}$ is a sequence of tokens $x_1 \dots x_n$ and $\boldsymbol{y}$ is a sequence of labels $y_1 \dots y_n$ where label $y_i \in \mathcal{Y}$ corresponds to token $x_i \in \mathcal{X}$. A key assumption in CRFs to keep learning and decoding efficient is the following.

Assumption 1.1. *The global feature function Φ is a sum of local functions ϕ. Throughout, we will assume that $\Phi(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^d$ has the form*

$$\Phi(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \phi(\boldsymbol{x}, i, y_{i-1}, y_i) \tag{2}$$

where $\phi(\boldsymbol{x}, i, y_{i-1}, y_i) \in \mathbb{R}^d$ is the local representation of $\boldsymbol{y}$ at position i.

2 Decoding

Suppose that we know the weight vector $\boldsymbol{w}$. Given an input sequence $\boldsymbol{x}$, how do we find a target sequence $\boldsymbol{y}^*$ such that $p(\boldsymbol{y}^*|\boldsymbol{x}) \geq p(\boldsymbol{y}|\boldsymbol{x})$ for all possible $\boldsymbol{y}$? Rather than calculating Eq. (1) for exponentially many candidates $\boldsymbol{y} \in \mathcal{Y}^n$, we make the following observation based on Assumption 1.1.

$$\begin{aligned}
\boldsymbol{y}^* &= \arg\max_{\boldsymbol{y}} \quad p(\boldsymbol{y}|\boldsymbol{x}) \\
&= \arg\max_{\boldsymbol{y}} \quad \frac{\exp(\boldsymbol{w} \cdot \Phi(\boldsymbol{x}, \boldsymbol{y}))}{\sum_{\boldsymbol{y}'} \exp(\boldsymbol{w} \cdot \Phi(\boldsymbol{x}, \boldsymbol{y}'))} \\
&= \arg\max_{\boldsymbol{y}} \quad \exp(\boldsymbol{w} \cdot \Phi(\boldsymbol{x}, \boldsymbol{y})) \\
&= \arg\max_{\boldsymbol{y}} \quad \boldsymbol{w} \cdot \Phi(\boldsymbol{x}, \boldsymbol{y}) \\
&= \arg\max_{\boldsymbol{y}} \quad \sum_i \boldsymbol{w} \cdot \phi(\boldsymbol{x}, i, y_{i-1}, y_i)
\end{aligned}$$

Note that if Φ is not decomposed, we cannot avoid explicitly enumerating $\boldsymbol{y}$. Thanks to the decomposition, however, a Viterbi algorithm can be used to find $\boldsymbol{y}^*$ such that

$$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} \sum_i \boldsymbol{w} \cdot \phi(\boldsymbol{x}, i, y_{i-1}, y_i)$$

In the algorithm, table π is used to record for each position i and state y

$$\pi[i, y] = \max_{\substack{y_1 \ldots y_i: \\ y_i = y}} \sum_{j=1}^{i} \boldsymbol{w} \cdot \phi(\boldsymbol{x}, j, y_{j-1}, y_j)$$

Then $\boldsymbol{y}^*$ can be retrieved from $\max_y \pi[n, y]$ using a backtracker κ.

Viterbi
Input: input sequence $\boldsymbol{x}$ of length n, weight vector $\boldsymbol{w}$
Output: optimal target sequence $\boldsymbol{y}^*$

- For all $y \in \mathcal{Y}$,

$$\pi[1, y] \leftarrow \boldsymbol{w} \cdot \phi(\boldsymbol{x}, 1, y_0, y) \text{ where } y_0 \text{ is a start symbol}$$

- For $i = 2 \ldots n$, for all $y \in \mathcal{Y}$,

$$\pi[i, y] \leftarrow \max_{y'} \pi[i-1, y'] + \boldsymbol{w} \cdot \phi(\boldsymbol{x}, i, y', y)$$
$$\kappa[i, y] \leftarrow \arg\max_{y'} \pi[i-1, y'] + \boldsymbol{w} \cdot \phi(\boldsymbol{x}, i, y', y)$$

- Return $\boldsymbol{y}^* = y_1^* \ldots y_n^*$ where

$$y_n^* \leftarrow \arg\max_{y} \pi[n, y]$$
$$y_i^* \leftarrow \kappa[i, y_{i+1}^*] \text{ for } i = n-1 \ldots 1$$

3 Learning

We calculate $\boldsymbol{w}$ that maximizes the regularized log likelihood of the training data $\{(\boldsymbol{x}^q, \boldsymbol{y}^q)\}_{q=1}^{Q}$ with some parameter $\lambda > 0$:

$$L(\boldsymbol{w}) = \sum_{q=1}^{Q} \log p(\boldsymbol{y}^q | \boldsymbol{x}^q) - \frac{\lambda}{2} ||\boldsymbol{w}||^2$$

$L(\boldsymbol{w})$ is a convex function in $\boldsymbol{w}$, so we can resort to some hill climbing method to find an optimal weight vector $\boldsymbol{w}^*$ such that

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} L(\boldsymbol{w})$$

Below is a gradient-based algorithm in its roughest form, but in practice one will have to be more clever.

GradientAscent
Input: training sequences $\{(\boldsymbol{x}^q, \boldsymbol{y}^q)\}_{q=1}^{Q}$, feature function $\Phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$
Output: weight vector $\boldsymbol{w}^* \in \mathbb{R}^d$ close to optimal

- $\boldsymbol{w} \leftarrow \mathbf{0}$
- Until convergence,
  - For $k = 1 \dots d$, calculate the gradient $\triangle_k \leftarrow \frac{\partial L(\boldsymbol{w})}{\partial w_k}$ and set

    $$w_k = w_k + \alpha \times \triangle_k$$

    where $\alpha > 0$ is some learning rate.
- Return $\boldsymbol{w}$.

We need to be able to compute the gradient of the function $L(\boldsymbol{w})$ with respect to each component w_k. Since CRFs are just log-linear models, we have the well-known form

$$\frac{\partial L(\boldsymbol{w})}{\partial w_k} = \sum_{q=1}^{Q} \Phi_k(\boldsymbol{x}^q, \boldsymbol{y}^q) - \sum_{q=1}^{Q} \sum_{\boldsymbol{y} \in \mathcal{Y}^n} p(\boldsymbol{y}|\boldsymbol{x}^q) \Phi_k(\boldsymbol{x}^q, \boldsymbol{y}) - \lambda w_k$$

The first and last terms are easy to compute. The middle term, which involves a sum over exponentially many sequences $\boldsymbol{y} \in \mathcal{Y}^n$, can be again computed efficiently by the following observation based on Assumption 1.1. For any $\boldsymbol{x} \in \mathcal{X}^n$,

$$\begin{aligned}
\sum_{\boldsymbol{y} \in \mathcal{Y}^n} p(\boldsymbol{y}|\boldsymbol{x}) \Phi_k(\boldsymbol{x}, \boldsymbol{y}) &= \sum_{\boldsymbol{y} \in \mathcal{Y}^n} p(\boldsymbol{y}|\boldsymbol{x}) \sum_{i=1}^{n} \phi_k(\boldsymbol{x}, i, y_{i-1}, y_i) \\
&= \sum_{i=1}^{n} \sum_{a,b \in \mathcal{Y}} \sum_{\substack{\boldsymbol{y} \in \mathcal{Y}^n: \\ y_{i-1}=a, y_i=b}} p(\boldsymbol{y}|\boldsymbol{x}) \phi_k(\boldsymbol{x}, i, y_{i-1}, y_i) \\
&= \sum_{i=1}^{n} \sum_{a,b \in \mathcal{Y}} \phi_k(\boldsymbol{x}, i, a, b) \sum_{\substack{\boldsymbol{y} \in \mathcal{Y}^n: \\ y_{i-1}=a, y_i=b}} p(\boldsymbol{y}|\boldsymbol{x}) \\
&= \sum_{i=1}^{n} \sum_{a,b \in \mathcal{Y}} \phi_k(\boldsymbol{x}, i, a, b) g(i, a, b|\boldsymbol{x})
\end{aligned}$$

where we define

$$g(i, a, b|\boldsymbol{x}) = \sum_{\substack{\boldsymbol{y} \in \mathcal{Y}^n: \\ y_{i-1}=a, y_i=b}} p(\boldsymbol{y}|\boldsymbol{x})$$

We can compute $g(i, a, b|\boldsymbol{x})$ for all $i \in [n]$ and $a, b \in \mathcal{Y}$ efficiently using a forward-backward algorithm. Thus we have shown that the gradient can be computed efficiently for training.

3.1 Forward-Backward

Given $\boldsymbol{x} \in \mathcal{X}^n$, we compute for every position $i \in [n]$ and label $y \in \mathcal{Y}$

$$\alpha[i, y] = \sum_{\substack{y_1 \cdots y_i: \\ y_i = y}} \prod_{j=1}^{i} \exp(\boldsymbol{w} \cdot \phi(\boldsymbol{x}, j, y_{j-1}, y_j))$$

$$\beta[i, y] = \sum_{\substack{y_i \cdots y_n: \\ y_i = y}} \prod_{j=i+1}^{n} \exp(\boldsymbol{w} \cdot \phi(\boldsymbol{x}, j, y_{j-1}, y_j))$$

Then it can be verified that

$$\begin{aligned} g(i, a, b|\boldsymbol{x}) &= \sum_{\substack{\boldsymbol{y} \in \mathcal{Y}^n: \\ y_{i-1} = a, y_i = b}} p(\boldsymbol{y}|\boldsymbol{x}) \\ &= \frac{\alpha[i-1, a] \times \exp(\boldsymbol{w} \cdot \phi(\boldsymbol{x}, i, a, b)) \times \beta[i, b]}{\sum_{y \in \mathcal{Y}} \alpha[n, y]} \end{aligned}$$

ForwardBackward
Input: input sequence $\boldsymbol{x}$ of length n
Output: $g(i, a, b|\boldsymbol{x})$ for all $i \in [n]$ and $a, b \in \mathcal{Y}$

- For all $y \in \mathcal{Y}$,

$$\alpha[1, y] \leftarrow \exp(\boldsymbol{w} \cdot \phi(\boldsymbol{x}, 1, y_0, y)) \text{ where } y_0 \text{ is a start symbol}$$

- For $i = 2 \ldots n$, for all $y \in \mathcal{Y}$,

$$\alpha[i, y] \leftarrow \sum_{y' \in \mathcal{Y}} \alpha[i-1, y'] \times \exp(\boldsymbol{w} \cdot \phi(\boldsymbol{x}, i, y', y))$$

- For all $y \in \mathcal{Y}$,

$$\beta[n, y] \leftarrow 1$$

- For $i = n - 1 \ldots 1$, for all $y \in \mathcal{Y}$,

$$\beta[i, y] \leftarrow \sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w} \cdot \phi(\boldsymbol{x}, i+1, y, y')) \times \beta[i+1, y']$$

- Return $g(i, a, b|\boldsymbol{x})$ where

$$g(i, a, b|\boldsymbol{x}) = \frac{\alpha[i-1, a] \times \exp(\boldsymbol{w} \cdot \phi(\boldsymbol{x}, i, a, b)) \times \beta[i, b]}{\sum_{y \in \mathcal{Y}} \alpha[n, y]}$$