10-801: Advanced Topics in Graphcal Models 10-801, Spring 2007

Conditional Random Fields

Lecturer: Eric P. Xing *Scribes: Ramesh Nallapati*

The objective is to predict the most likely sequence of labels $\hat{\mathbf{y}}$ given a data sequence $\mathbf{x}$:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) \tag{1}$$

In generative models such as HMM, we use Bayes rule to compute the conditional likelihood of labels:

$$\arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y}} \frac{P(\mathbf{x},\mathbf{y})}{P(\mathbf{x})} = \arg\max_{\mathbf{y}} P(\mathbf{x},\mathbf{y}) \tag{2}$$

The objective function optimized by HMM is the joint likelihood, which is different from what we want to optimize. As a solution, discriminative models called MEMMs were proposed which optimize the conditional likelihood of the labels given the observed data directly:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y}} \prod_{i=1}^{n} P(y_i|y_{i-1}, x_i) \tag{3}$$

where i is the position in the sequence of size n. However, MEMMs suffer from label bias problem, i.e., local distributions influence the global choice of latent variables as shown in figure 1. In this figure, the model is forced to choose state $y^{(2)}$ correspondong to observation x_2, although the evidence points to state $y^{(1)}$. This follows from the fact that the probabilities in MEMMs are locally normalized, and as a result, it forces the model to prefer state $y^{(2)}$ in this case, since it has a higher incoming-mass from the previous states.

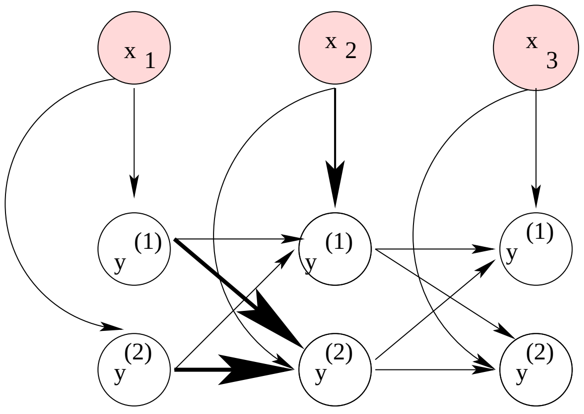

Figure 1: label bias problem with MEMMs: the thickness of the arrow indicates the magnitude of the probability mass

One way to resolve this problem is to normalize the probabilities globally. The resulting model is called a *Conditional Random Field*, which is shown below.

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{i=1}^{n}\sum_{k=1}^{K} \theta_k f_k(y_i, y_{i-1}, \mathbf{x}, i)\right) \tag{4}$$

where K is the number of feature functions and i is the position index in the example as usual.

1 Inference

Given a trained model, inference on a test example corresponds to finding the most likely label sequence.

$$\begin{aligned}
\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \log P(\mathbf{y}|\mathbf{x}) &= \arg\max_{\mathbf{y}} \left(\sum_{i=1}^{n} \sum_{k=1}^{K} \theta_k f_k(y_i, y_{i-1}, \mathbf{x}, i) - \log Z(\mathbf{x}) \right) \\
&= \arg\max_{\mathbf{y}} \sum_{i=1}^{n} \sum_{k=1}^{K} \theta_k f_k(y_i, y_{i-1}, \mathbf{x}, i) \qquad (5)
\end{aligned}$$

This can be easily done using the standard Viterbi decoding used in HMMs.

2 Learning

Define $F_k(\mathbf{y}, \mathbf{x}) = \sum_i f_k(y_i, y_{i-1}, \mathbf{x}, i)$ and the column vectors $\mathbf{F}(\mathbf{y}, \mathbf{x}) = (F_1(\mathbf{y}, \mathbf{x}), \cdots, F_K(\mathbf{y}, \mathbf{x}))^T$ and $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_K)^T$.

Now the conditional log-likelihood of N training examples $\{(\mathbf{x}_1, \mathbf{y}_1), \cdots, (\mathbf{x}_N, \mathbf{y}_N)\}$ is given by:

$$\sum_{j=1}^{N} \log P_{\boldsymbol{\theta}}(\mathbf{y}_j|\mathbf{x}_j) = \sum_{j} \left(\boldsymbol{\theta}^T \mathbf{F}(\mathbf{y}_j, \mathbf{x}_j) - \log(\sum_{\mathbf{y}} \exp(\boldsymbol{\theta}^T \mathbf{F}(\mathbf{y}, \mathbf{x}_j)) \right) \qquad (6)$$

Training consists of estimating the parameters $\boldsymbol{\theta}$ that optimize the conditional likelihood defined above. Its first derivative is given by:

$$\begin{aligned}
\nabla(\sum_{j=1}^{N} \log P_{\boldsymbol{\theta}}(\mathbf{y}_j|\mathbf{x}_j)) &= \sum_{j} \left(\mathbf{F}(\mathbf{y}_j, \mathbf{x}_j) - \frac{\sum_{\mathbf{y}} \exp(\boldsymbol{\theta}^T \mathbf{F}(\mathbf{y}, \mathbf{x}_j) \mathbf{F}(\mathbf{y}, \mathbf{x}_j)}{\sum_{\mathbf{y}} \exp(\boldsymbol{\theta}^T \mathbf{F}(\mathbf{y}, \mathbf{x}_j)} \right) \\
&= \sum_{j} \left(\mathbf{F}(\mathbf{y}_j, \mathbf{x}_j) - E_{P_{\boldsymbol{\theta}}(\mathbf{Y}|\mathbf{x}_j)}[F(\mathbf{Y}, \mathbf{x}_j)] \right) \qquad (7)
\end{aligned}$$

where j is the index of the training example.

2.1 Computing Z(x)

For a given $\mathbf{x}$, $Z(\mathbf{x})$ can be efficiently computed as follows.

$$\begin{aligned}
Z(\mathbf{x}) &= \sum_{\mathbf{y}} \left\{ \exp(\sum_{k=1}^{K} \theta_k f_k(y_1, y_{start}, \mathbf{x}, 1)) \exp(\sum_{i=2}^{n} \sum_{k=1}^{K} \theta_k f_k(y_i, y_{i-1}, \mathbf{x}, i)) \exp(\sum_{k=1}^{K} \theta_k f_k(y_{end}, y_n, \mathbf{x}, n+1)) \right\} \\
&= \sum_{\mathbf{y}} \left\{ M_1(y_1, start) \prod_{i=2}^{n} M_i(y_i, y_{i-1}, \mathbf{x}) M_{n+1}(y_{end}, y_n, \mathbf{x}, n+1) \right\} \\
&\qquad \text{where } M_i(y_i, y_{i-1}, \mathbf{x}) = \exp(\sum_{k} \theta_k f_k(y_i, y_{i-1}, \mathbf{x}, i)) \\
&= [\prod_{i=1}^{n+1} M_i(\mathbf{x})]_{start,end} \text{ where } M_i(\mathbf{x}) \text{ is a } |Y| \times |Y| \text{ matrix with } M_i(\mathbf{x})_{y,y'} = M_i(y_i = y, y_{i-1} = y', \mathbf{x}) \qquad (8)
\end{aligned}$$

where *start* and *end* are artificial states defined to indicate the beginning and end of a sequence.

2.2 Computing $E_{P_{\boldsymbol{\theta}}(\mathbf{Y}|\mathbf{x}_j)}[F(\mathbf{Y}, \mathbf{x}_j)]$

First we note that

$$E_{P_{\boldsymbol{\theta}}(\mathbf{Y}|\mathbf{x}_j)}[F(\mathbf{Y}, \mathbf{x}_j)] = \frac{1}{Z(\mathbf{x}_j)} \sum_{\mathbf{y}} \exp(\boldsymbol{\theta}^T \mathbf{F}(\mathbf{y}, \mathbf{x}_j) \mathbf{F}(\mathbf{y}, \mathbf{x}_j) \tag{9}$$

Since we already know how to compute $Z(\mathbf{x}_j)$ efficiently, we focus on $\sum_{\mathbf{y}} \exp(\boldsymbol{\theta}^T \mathbf{F}(\mathbf{y}, \mathbf{x}_j) \mathbf{F}(\mathbf{y}, \mathbf{x}_j)$, in partiuclar, one of its components $\sum_{\mathbf{y}} \exp(\boldsymbol{\theta}^T \mathbf{F}(\mathbf{y}, \mathbf{x}_j) F_k(\mathbf{y}, \mathbf{x}_j)$. We can express it as follows.

$$\begin{aligned}
\sum_{\mathbf{y}} \exp(\boldsymbol{\theta}^T \mathbf{F}(\mathbf{y}, \mathbf{x}_j) F_k(\mathbf{y}, \mathbf{x}_j) &= \sum_{\mathbf{y}} \exp(\sum_{i=1}^{n+1} \sum_{k=1}^{K} \theta_k f_k(y_i, y_{i-1}, \mathbf{x}, i)) F_k(\mathbf{y}, \mathbf{x}_j) \\
&= \sum_{\mathbf{y}} \exp(\sum_{i=1}^{n+1} \sum_{k=1}^{K} \theta_k f_k(y_i, y_{i-1}, \mathbf{x}, i)) \sum_{i=1}^{n+1} f_k(y_i, y_{i-1}, \mathbf{x}_j) \qquad (10)
\end{aligned}$$

Using the result from Eq. (8), it is easy to see that we can rewrite Eq. (10) as follows.

$$\begin{aligned}
\sum_{\mathbf{y}} \exp(\boldsymbol{\theta}^T \mathbf{F}(\mathbf{y}, \mathbf{x}_j) F_k(\mathbf{y}, \mathbf{x}_j) &= \sum_{i} \sum_{y', y} [\prod_{l=1}^{i-1} M_l(\mathbf{x})]_{start, y'} M_i(y', y, \mathbf{x}) f_k(y, y', \mathbf{x}, i) [\prod_{m=i+1}^{n+1} M_m(\mathbf{x})]_{y, end} \\
&= \sum_{i} \sum_{y', y} \alpha_{i-1}(y'|\mathbf{x}) M_i(y', y, \mathbf{x}) f_k(y, y', \mathbf{x}, i) \beta_i(y|\mathbf{x}) \qquad (11)
\end{aligned}$$

where we define the forward and backward vectors α and β recursively as follows.

$$\begin{aligned}
[\alpha_0(\mathbf{x})]_y &= 1 \text{ if } y = \mathit{start} \\
&= 0 \text{ otherwise} \qquad (12) \\
\alpha_i(\mathbf{x}) &= (\alpha_{i-1}(\mathbf{x}))^T M_i(\mathbf{x}) \qquad (13)
\end{aligned}$$

Similarly,

$$\begin{aligned}
[\beta_{n+1}(\mathbf{x})]_y &= 1 \text{ if } y = \mathit{end} \\
&= 0 \text{ otherwise} \qquad (14) \\
\beta_i(\mathbf{x}) &= M_{i+1}(\mathbf{x}) \beta_{i+1}(\mathbf{x}) \qquad (15)
\end{aligned}$$

Thus we can compute the gradient efficiently using the forward backward vectors. One can then arrive at the optimum solution by using an efficient pre-conditioner based conjugate gradient method or a limited memory Quasi-Newton method as described in (Sha and Pereira, 2006).