CRF gradients without forward passes

Charles Sutton

April 7, 2006

Typically, the gradient of a linear-chain CRF is calculated using $\xi_t(i,j) = p(y_t = i, y_{t+1} = j|\mathbf{x})$, which requires both a forward and a backward pass. But the gradient can be computed using only backward passes, and the resulting algorithm is essentially backpropagation, but using the log loss rather than the squared error.

Let the CRF be defined as

$$p(\mathbf{y}|\mathbf{x}) = Z(\mathbf{x})^{-1}\psi_0(y_0, \mathbf{x}_0)\prod_t \psi_t(y_t, y_{t+1}, \mathbf{x}_t),$$

where the factors are

$$\psi_t(y_t, y_{t+1}, \mathbf{x}_t) = \exp\{\sum_k \lambda_k f_k(y_t, y_{t+1}, \mathbf{x}_t)\}.$$

The trick is to recursively compute the gradient of $\log \beta_t(j)$ using a second backward pass. That is, since

$$\log \beta_t(i) = \log \sum_j \exp\left\{\log \psi_t(i,j) + \log \beta_{t+1}(j)\right\}, \tag{1}$$

we can recursively compute for each CRF weight λ_k

$$\frac{\partial \log \beta_t(i)}{\partial \lambda_k} = \frac{1}{\beta_t(i)} \sum_j \frac{\partial}{\partial \lambda_k} \left\{\exp(\log \psi_t(i,j) + \log \beta_{t+1}(j))\right\} \tag{2}$$

$$= \frac{1}{\beta_t(i)} \sum_j \exp(\log \psi_t(i,j) + \log \beta_{t+1}(j)) \left[f_k(i,j,\mathbf{x}_t) + \frac{\partial \log \beta_{t+1}(j)}{\partial \lambda_k}\right] \tag{3}$$

$$= \sum_j p(y_{t+1} = j|y_t = i, \mathbf{x}) \left[f_k(i,j,\mathbf{x}_t) + \frac{\partial \log \beta_{t+1}(j)}{\partial \lambda_k}\right] \tag{4}$$

Finally, if ψ_0 are the initial costs, then

$$\log Z(\mathbf{x}) = \sum_s \exp\{\log \psi_0(s, \mathbf{x}_0) + \log \beta_0(s)\}, \tag{5}$$

which means we can calculate the derivative we need for the likelihood gradient:

$$\frac{\partial \log Z(\mathbf{x})}{\partial \lambda_k} = \frac{1}{Z(\mathbf{x})} \sum_i \exp\{\log \psi_0(s, \mathbf{x}_0) + \log \beta_0(s)\} \left(\frac{\partial \log \psi_0(s, \mathbf{x}_0)}{\partial \lambda_k} + \frac{\partial \log \beta_0(i)}{\partial \lambda_k}\right) \tag{6}$$

$$= \sum_i p(y_0 = i|\mathbf{x}) \left(\frac{\partial \log \psi_0(s, \mathbf{x}_0)}{\partial \lambda_k} + \frac{\partial \log \beta_0(i)}{\partial \lambda_k}\right). \tag{7}$$

Thus, we can calculate the CRF gradient using two sets of backward recursions, rather than a backward and a forward recursion. This technique is very reminiscent of backpropagation, but on the log-loss rather than the squared error. This algorithm calculates the gradient exactly, so of course you don't lose convexity when you do this. One presumes that a generalized version of this would be applicable to models with latent variables, just as in neural networks.

To implement this, observe that you need a separate backward recursion for every feature in the model. Since standard forward-backward requires two recursive passes, it is not clear if this backward-only technique has any advantages in practice. An initial implementation of this appeared to be slower than standard forward-backward.

Acknowledgments

This note was inspired by a conversation with John Lafferty.