CRF Tutorial

zenRRan

April 22, 2018

1 概述

CRF（Conditional Random Field），中文被翻译为条件随机场。经常被用于序列标注，其中包括词性标注，分词，命名实体识别等领域。但是为什么叫这个名字呢？下面看完了基本也就明白了！那我们继续吧。

2 理论

我们以词性标注为例，先介绍下词性标注的概念：

- $instance$:我 去 北京
- $word\_alphabet$: {0:'我',1:'去',2:'北京'}
- $label\_alphabet$: {0:'PN',1:'VV',2:'NN'}
- $word\_nums$: 3
- $label\_nums$: 3

这个表示 词:词性，分别为 我:PN，去:VV，北京:NN。

Table 1: 例子

$label\_index$	0	1	2
$label$	PN	VV	NN
$word$	我	去	北京
$word\_index$	0	1	2

$Table1$就是$word$和$label$数字化后变成$word_index$，$label\_index$。最终就变成$Table2$的形式：

Table 2: 数字化

$label\_index$	0	1	2
$word\_index$	0	1	2

上述是标准金标，也就是正确答案，但是实际上电脑预测的不会是正确的。因为$label$有3种，每一个字被预测的$label$就有3种可能，为了数字化这些可能，我们从$word_index$ 到$label_index$ 设置一种分数，叫做发射分数$emit$，简化为E。

$word\_index$ 的2到$label_index$的2，像不像发射上去的？此时的分数就记作发射分数$E[2][2]$。

另外，我们想想，如果单单就这个发射分数来评价，太过于单一了，因为这个是一个序列，比如前面的$label$为1代表VV动词，那此时的$label$被预测的肯定不能是VV,因为动词后面不能接动词，所以知道前一个$label$转向后一个$label$可能性会增加准确率，所以这个时候就需要一个分数代表前一个$label$ 到此时$label$ 的分数，我们叫这个为转移分数，即T。如图，横向的$label$到$label$，就是由一个$label$到另一个$label$转移的意思，此时的分数为$T[1][1]$。

假设$word_index = i$到$label_index = j$的分数为$s[i][j]$，则

$$s[0][0] = E[0][0]$$

因为$word_index = 0$前面没有$word_index$了，所以$s[0][0]$就为发射分数$E[0][0]$。$word_index = 1$到$label_index = 1$的分数$s[1][1]$为$E[1][1]$+$T[0][1]$。但是CRF为了全局考虑，将前一个的分数也累加到当前分数上，这样更能表达出已经预测的序列的整体分数，则：

$$s[1][1] = s[0][0] + E[1][1] + T[0][1]$$

所以，$s[2][2]$也就很容易了：

$$s[2][2] = E[0][0] + E[1][1] + T[0][1] + E[2][2] + T[1][2]$$

因为$s[2][2]$已经为最后的词的的分数，所以该$s[2][2]$为金标$score$({我 去 北京}, {PN VV NN})即$score$({0 1 2}, {0 1 2})的最终得分。最后的公式总结为：

$$score(X, y) = \sum_{i=0}^{n} T_{y_i, y_{i+1}} + \sum_{i=0}^{n} E_{i, y_i} \qquad (1)$$

其中X为$word_index$序列，y为预测的$label_index$序列。

因为这个预测序列有很多种，种类为$label$的可重复排列组合大小。其中只有一种组合是对的，我们只想通过神经网络训练使得对的$score$ 的比重在总体的所有$score$的越大越好。而这个时候我们一般$softmax$化，即：

$$p(y_{gold}|X) = \frac{e^{s(X,y_{gold})}}{\sum_{y \in Y_x} e^{s(X,y)}}$$

其中分子中的s为$label$序列为正确序列的$score$，分母为每种可能的$score$的总和。

这个比值越大，我们的预测就越准，所以，这个公式也就可以当做我们的$loss$，可是$loss$一般都越小越好，那我们就对这个加个负号即可，但是这个最终结果是趋近于1的，我们实验的结果是趋近于0的，这时候log就派上用场了，即：

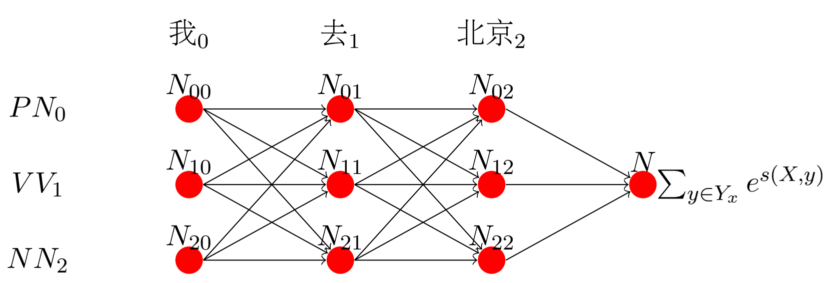
$$-log(p(y_{gold}|X)) = -log(\frac{e^{score(X,y_{gold})}}{\sum_{y \in Y_x} e^{score(X,y)}})$$

$$= -score(X, y_{gold}) + log(\sum_{y \in Y_x} e^{score(X,y)})$$

当然这个log也有平滑结果的功效。

3 计算所有路径的得分

$loss$的分子在上面已经求出来了，现在就差分母了，而计算所有预测序列可能的得分和也就是计算所有路径的得分。我们第一种想法就是每一种可能都求出来，然后累加即可。可是，比如$word$序列长为10，$label$种类为7，那么总共需要计算10^7次，这样的计算太耗时间了。那么怎么计算的时间快呢？这里有一种方法，就是每个节点记录之前所有节点到当前节点的路径总和。如图：



注明：下面的$N[i][j]$都为上图的N_{ij}。

解释下这个图：

第一列：

首先说下，因为‘我’是第一列，前面没有别的词，所以就不用加上前面的值。继续说，$N[0][0]$表示‘我’选择PN的得分，$N[1][0]$表示‘我’选择VV的得分,$N[2][0]$表示‘我’选择NN的得分而该得分只有发射得分，所以为：

$$N[0][0] = E[0][0]$$

同理，得：

$$N[1][0] = E[0][1]$$

$$N[2][0] = E[0][2]$$

再来分析第二列:

$N[0][1]$表示前一个选择PN的得分+‘去’选择PN的得分（‘去’选择PN的得分为T[0][0]+E[1][0]），前一个选择VV的得分+加上‘去’选择PN 的得分，加上前一个选择NN的得分+‘去’选择PN的得分。公式为：

$$N[0][1] = log(e^{N[0][0]+T[0][0]+E[1][0]} + e^{N[1][0]+T[1][0]+E[1][0]} + e^{N[2][0]+T[2][0]+E[1][0]})$$

类推：

$$N[1][1] = log(e^{N[0][0]+T[0][1]+E[1][1]} + e^{N[1][0]+T[1][1]+E[1][1]} + e^{N[2][0]+T[2][1]+E[1][1]})$$

$$N[2][1] = log(e^{N[0][0]+T[0][2]+E[1][2]} + e^{N[1][0]+T[1][2]+E[1][2]} + e^{N[2][0]+T[2][2]+E[1][2]})$$

再类推第三列：

$$N[0][2] = log(e^{N[0][1]+T[0][0]+E[2][0]} + e^{N[1][1]+T[1][0]+E[2][0]} + e^{N[2][1]+T[2][0]+E[2][0]})$$

$$N[1][2] = log(e^{n[0][1]+T[0][1]+E[2][1]} + e^{n[1][1]+T[1][1]+E[2][1]} + e^{n[2][1]+T[2][1]+E[2][1]})$$

$$N[2][2] = log(e^{N[0][1]+T[0][2]+E[2][2]} + e^{N[1][1]+T[1][2]+E[2][2]} + e^{N[2][1]+T[2][2]+E[2][2]})$$

最后一列求完了，因为每个节点都包含了该节点之前所有节点到该节点的可能路径，因为现在的

$$N = log(e^{N[0][2]} + e^{N[1][2]} + e^{N[2][2]})$$

的总和就是所有路径的总和，也就是我们要求的损失函数里面的$\sum_{y \in Y_x} e^{s(X,y)}$，即为：

$$\sum_{y \in Y_x} e^{s(X,y)} = N$$

4 得出具体损失函数

最终的我们的损失函数求出来了：

$$-log(p(y_{gold}|X)) = -s(X,y) + log(\sum_{y \in Y_x} e^{s(X,y)})$$

$$= -s[2][2] + log(N)$$

这样我们就能根据损失函数反向传播梯度，更新T E参数了。

5 batch

上面的那种求总和的方法，还有一种好处就是可以加快并行计算，也就刚好能做多个句子的$batch$批处理。先说什么是并行计算，字面意思就能理解，并行，并排行进，大家同时进行的意思，同时进行的前提条件是需要用到的东西都已经准备好。放在计算机里的意思就是当前运行的程序需要的数据都已经准备好了。那我们来看看我们的数据怎么能并行计算吧，我拿出来一列数据来看看（先说下为什么拿出的是一列，而不是一行，因为一列所需要的数据前一列都已经计算过了，而一行不具备这样的条件），比如第二列：

$$N[0][1] = log(e^{N[0][0]+T[0][0]+E[1][0]} + e^{N[1][0]+T[1][0]+E[1][0]} + e^{N[2][0]+T[2][0]+E[1][0]})$$

$$N[1][1] = log(e^{N[0][0]+T[0][1]+E[1][1]} + e^{N[1][0]+T[1][1]+E[1][1]} + e^{N[2][0]+T[2][1]+E[1][1]})$$

$$N[2][1] = log(e^{N[0][0]+T[0][2]+E[1][2]} + e^{N[1][0]+T[1][2]+E[1][2]} + e^{N[2][0]+T[2][2]+E[1][2]})$$

但是这样或许看不到什么效果，我来整理下，去掉log，去掉e，只提取数据：

$N[0][1]\ data:$

$$N[0][0] + T[0][0] + E[1][0]$$

$$N[1][0] + T[1][0] + E[1][0]$$

$$N[2][0] + T[2][0] + E[1][0]$$

$N[1][1]\ data:$

$$N[0][0] + T[0][1] + E[1][1]$$

$$N[1][0] + T[1][1] + E[1][1]$$

$$N[2][0] + T[2][1] + E[1][1]$$

$N[2][1]\ data:$

$$N[0][0] + T[0][2] + E[1][2]$$

$$N[1][0]+T[1][2]+E[1][2]$$
$$N[2][0]+T[2][2]+E[1][2]$$

我们先看N这三组数据，发现每组里面N都是一样都为$N[0][0], N[1][0], N[2][0]$，所以我们可以设定矩阵$N$为：

$$\begin{bmatrix} 00 \\ 10 \\ 20 \end{bmatrix}$$

我们看到矩阵N第0维循环变化，第1维不变，但是上面的只是一组数据，我们需要三组，所以我们对该N进行二维扩展，也就是列复制：

$$\begin{bmatrix} 00 & 00 & 00 \\ 10 & 10 & 10 \\ 20 & 20 & 20 \end{bmatrix}$$

再看T，第一组为$T[0][0], T[1][0], T[2][0]$，第二组为$T[0][1], T[1][1], T[2][1]$，第三组为$T[0][2], T[1][2], T[2][2]$。我们能其实够很明显的看出第一组为$T$的$3*3$矩阵第0列，剩下的分别为第1列，第2列，即矩阵$T$为：

$$\begin{bmatrix} 00 & 01 & 02 \\ 10 & 11 & 12 \\ 20 & 21 & 22 \end{bmatrix}$$

最后同理我们看E，我们会观察到它和N的情况相似，但是E第0维不变，第1维是循环变化，所以是行复制：

$$\begin{bmatrix} 10 & 11 & 12 \\ 10 & 11 & 12 \\ 10 & 11 & 12 \end{bmatrix}$$

我们会发现，矩阵$N\ T\ E$的第一列按位相加的结果刚好是$N[0][1]$，同理的

第二列，第三列分别按位相加分别得$N[1][1]$，$N[2][1]$，即：

$$\begin{bmatrix} N_{00} & N_{00} & N_{00} \\ N_{10} & N_{10} & N_{10} \\ N_{20} & N_{20} & N_{20} \end{bmatrix} + \begin{bmatrix} T_{00} & T_{01} & T_{02} \\ T_{10} & T_{11} & T_{12} \\ T_{20} & T_{21} & T_{22} \end{bmatrix} + \begin{bmatrix} E_{10} & E_{11} & E_{12} \\ E_{10} & E_{11} & E_{12} \\ E_{10} & E_{11} & E_{12} \end{bmatrix}$$
$$= \begin{bmatrix} N_{00}+T_{00}+E_{10} & N_{00}+T_{01}+E_{11} & N_{00}+T_{01}+E_{12} \\ N_{10}+T_{10}+E_{10} & N_{10}+T_{11}+E_{11} & N_{10}+T_{11}+E_{12} \\ N_{20}+T_{20}+E_{10} & N_{20}+T_{21}+E_{11} & N_{20}+T_{21}+E_{12} \end{bmatrix}$$
$$replace -> \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix}$$
$$add\ e -> \begin{bmatrix} e^{a_{00}} & e^{a_{01}} & e^{a_{02}} \\ e^{a_{10}} & e^{a_{11}} & e^{a_{12}} \\ e^{a_{20}} & e^{a_{21}} & e^{a_{22}} \end{bmatrix}$$
$$add\ log\_sum ->$$
$$\begin{bmatrix} log(e^{a_{00}}+e^{a_{10}}+e^{a_{20}}) & log(e^{a_{01}}+e^{a_{11}}+e^{a_{21}}) & log(e^{a_{02}}+e^{a_{12}}+e^{a_{22}}) \end{bmatrix}$$
$$= \begin{bmatrix} N_{01} & N_{11} & N_{21} \end{bmatrix}$$

同理，求出$N_{02}\ N_{12}\ N_{22}$，然后

$$log(\sum_{y \in Y_x} e^{s(X,y)}) = log(e^{N_{02}} + e^{N_{12}} + e^{N_{22}})$$

上面的只是表示一个句子的计算，我们为了加快速度，或者使用GPU的时候，需要用到$batch$，那么$batch$里的上述$N\ T\ E$是怎么个存在形式呢？以$batch = n$为例：N数据格式为：

$$\begin{bmatrix} \begin{bmatrix} N_{00}^0 & N_{00}^0 & N_{00}^0 \\ N_{10}^0 & N_{10}^0 & N_{10}^0 \\ N_{20}^0 & N_{20}^0 & N_{20}^0 \end{bmatrix} \begin{bmatrix} N_{00}^1 & N_{00}^1 & N_{00}^1 \\ N_{10}^1 & N_{10}^1 & N_{10}^1 \\ N_{20}^1 & N_{20}^1 & N_{20}^1 \end{bmatrix} \ldots \begin{bmatrix} N_{00}^{n-1} & N_{00}^{n-1} & N_{00}^{n-1} \\ N_{10}^{n-1} & N_{10}^{n-1} & N_{10}^{n-1} \\ N_{20}^{n-1} & N_{20}^{n-1} & N_{20}^{n-1} \end{bmatrix} \end{bmatrix}$$

T数据格式为：

$$\begin{bmatrix} \begin{bmatrix} T_{00}^0 & T_{01}^0 & T_{02}^0 \\ T_{10}^0 & T_{11}^0 & T_{12}^0 \\ T_{20}^0 & T_{21}^0 & T_{22}^0 \end{bmatrix} \begin{bmatrix} T_{00}^1 & T_{01}^1 & T_{02}^1 \\ T_{10}^1 & T_{11}^1 & T_{12}^1 \\ T_{20}^1 & T_{21}^1 & T_{22}^1 \end{bmatrix} \ldots \begin{bmatrix} T_{00}^{n-1} & T_{01}^{n-1} & T_{02}^{n-1} \\ T_{10}^{n-1} & T_{11}^{n-1} & T_{12}^{n-1} \\ T_{20}^{n-1} & T_{21}^{n-1} & T_{22}^{n-1} \end{bmatrix} \end{bmatrix}$$

E数据格式为：

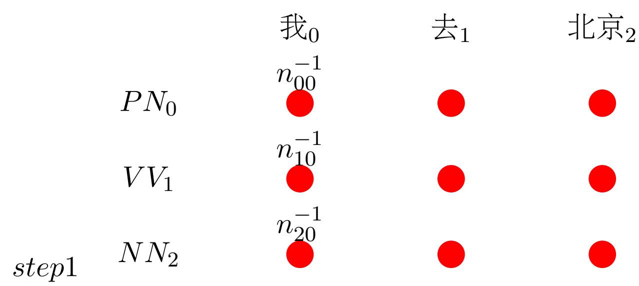
$$\begin{bmatrix} \begin{bmatrix} E_{10}^0 & E_{11}^0 & E_{12}^0 \\ E_{10}^0 & E_{11}^0 & E_{12}^0 \\ E_{10}^0 & E_{11}^0 & E_{12}^0 \end{bmatrix} \begin{bmatrix} E_{10}^1 & E_{11}^1 & E_{12}^1 \\ E_{10}^1 & E_{11}^1 & E_{12}^1 \\ E_{10}^1 & E_{11}^1 & E_{12}^1 \end{bmatrix} \ldots \begin{bmatrix} E_{10}^{n-1} & E_{11}^{n-1} & E_{12}^{n-1} \\ E_{10}^{n-1} & E_{11}^{n-1} & E_{12}^{n-1} \\ E_{10}^{n-1} & E_{11}^{n-1} & E_{12}^{n-1} \end{bmatrix} \end{bmatrix}$$

其中，X_j^i中的i表示$batch$里的第i组矩阵，j表示$batch$里的第i组中位置为j的数据。

6 预测过程

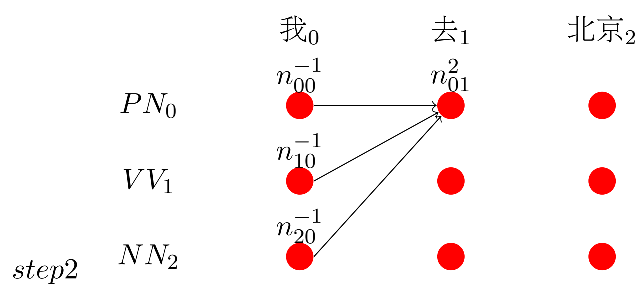
上面是Encoder编码过程，训练完了，该看看训练的效果了，这里预测的过程叫做Decoder解码过程。这时候N E T都是固定了的，不会再变化。我们的目的是，选取可能性最高的，又因为可能性最高在这里表示得分最高，然后根据最高的得分，我们向前一个一个的选取每次前一个最高得分的节点，最终这些所有的节点就是我们的最后的预测序列。这个过程是不是很熟悉的感觉，对就是我们的动态规划算法，但是在这里叫维特比算法。我们来走一遍过程：

每个节点选取得分最高的路径并记录得分和选的哪条路径：其中n_{ij}^s中的s表示前一条路径，没有的就是-1，n_{ij}表示前节点到当前节点的最佳得分。 此时

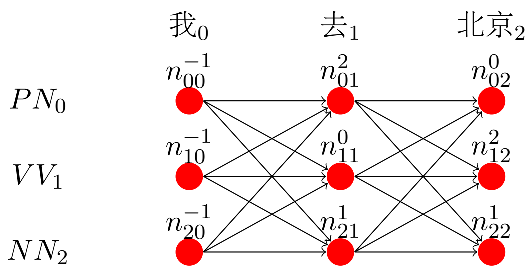


$$n_{01} = \max \begin{cases} n_{00} + E[1][0] + T[0][0] \\ n_{10} + E[1][0] + T[1][0] \\ n_{20} + E[1][0] + T[2][0] \end{cases}$$

比如此时预测$n_{20} + E[1][0] + T[2][0]$为最高的，则记录$n_{01} = n_{20} + E[1][0] + T[2][0]$且路径为2,综上记为$n_{01}^2$:

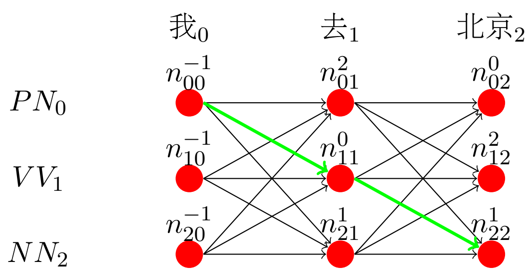


同理，我们假设有了$step3$的最终结果。



$step3$

在最后我们假如n_{22}最大，而且它的前一个路径为1，则看到n_{11}的前一个路径为0，而且n_{00}的前一个路径为-1，表示结束，则整个路径就有了，即为$n_{00}->n_{11}->n_{22}$，如图$step4$:



$step4$

由$step4$得，最终('我'，'去'，'北京')的预测结果为：('我'$->PN$，'去'$->VV$，'北京'$->NN$)。