PJ2 Part2实验报告

CRF

CRF 是指条件随机场（Conditional Random Field），是给定一组输入随机变量条件下另一组输出随机变量的条件概率分布模型，其特点是假设输出随机变量构成马尔可夫随机场。条件随机场**可以用于解决标注问题。**我们的任务涉及到的是**线性链(limear chain)条件随机场**，问题为由输入序列对输出序列预测的判别模型，形式为对数线性模型，其学习方法通常是极大似然估计或正则化的极大似然估计。

条件随机场的定义如下：

> **定义 11.3（条件随机场）** 设 X 与 Y 是随机变量，$P(Y|X)$ 是在给定 X 的条件下 Y 的条件概率分布。若随机变量 Y 构成一个由无向图 $G=(V,E)$ 表示的马尔可夫随机场，即
>
> $$P(Y_v|X,Y_w,w\neq v)=P(Y_v|X,Y_w,w\sim v) \tag{11.8}$$
>
> 对任意结点 v 成立，则称条件概率分布 $P(Y|X)$ 为条件随机场。式中 $w\sim v$ 表示在图 $G=(V,E)$ 中与结点 v 有边连接的所有结点 w，$w\neq v$ 表示结点 v 以外的所有结点，Y_v，Y_u 与 Y_w 为结点 v，u 与 w 对应的随机变量。

这里解释一下无向图表示的马尔可夫随机场：

如果联合概率分布P(Y) 满足成对、局部、全局马尔可夫性，则称此联合概率分布为马尔可夫随机场。

成对马尔可夫性：

> 成对马尔可夫性：设 u 和 v 是无向图 G 中任意两个没有边连接的结点，结点 u 和 v 分别对应随机变量 Y_u 和 Y_v。其他所有结点为 O，对应的随机变量组是 Y_O。成对马尔可夫性是指给定随机变量组 Y_O 的条件下随机变量 Y_u 和 Y_v 是条件独立的，即
>
> $$P(Y_u,Y_v|Y_O)=P(Y_u|Y_O)P(Y_v|Y_O) \tag{11.1}$$

局部马尔可夫性：

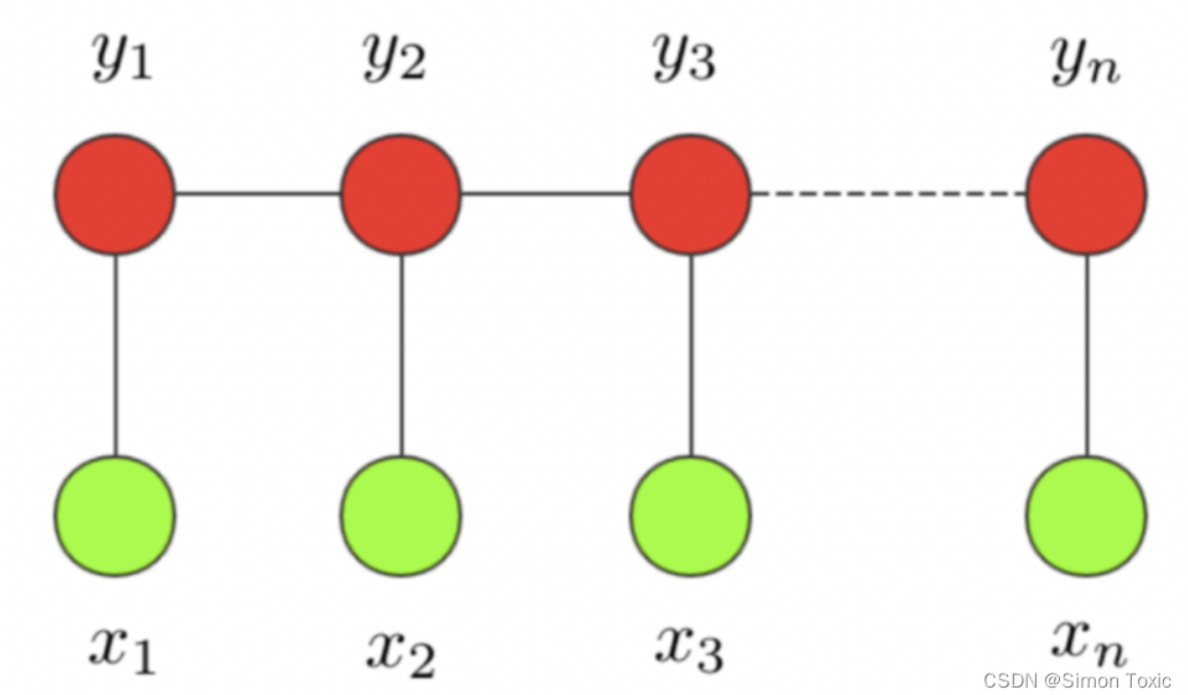
> 局部马尔可夫性：设 $v\in V$ 是无向图 G 中任意一个结点，W 是与 v 有边连接的所有结点，O 是 v 和 W 以外的其他所有结点。v 表示的随机变量是 Y_v，W 表示的随机变量组是 Y_W，O 表示的随机变量组是 Y_O。局部马尔可夫性是指在给定随机变量组 Y_W 的条件下随机变量 Y_v 与随机变量组 Y_O 是独立的，即
>
> $$P(Y_v,Y_O|Y_W)=P(Y_v|Y_W)P(Y_O|Y_W) \tag{11.2}$$
>
> 在 $P(Y_O|Y_W)>0$ 时，等价地，
>
> $$P(Y_v|Y_W)=P(Y_v|Y_W,Y_O) \tag{11.3}$$

全局马尔可夫性：

全局马尔可夫性：设结点集合 A，B 是在无向图 G 中被结点集合 C 分开的任意结点集合，如图 11.2 所示。结点集合 A，B 和 C 所对应的随机变量组分别是 Y_A，Y_B 和 Y_C。全局马尔可夫性是指给定随机变量组 Y_C 条件下随机变量组 Y_A 和 Y_B 是条件独立的，即

$$P(Y_A, Y_B|Y_C) = P(Y_A|Y_C)P(Y_B|Y_C) \tag{11.4}$$

三种马尔可夫性的定义是等价的，我的理解是，这三个马尔可夫性说明的都是同一点：**联合概率分布中的点只受和它有关联的点的影响，与和它无关联的点无关。**所以条件随机场中也会提及团、最大团的概念，因为团中任意两个点都是相连的，他们之间是由概率影响的。结合下图理解：



CRF 本质是一个无向图，其中绿色点表示输入，红色点表示输出。点与点之间的边可以分成两类，一类是 x 与 y 之间的连线，表示其相关性；另一类是相邻时刻的 y之间的相关性。也就是说，在预测某时刻 y 时，**同时要考虑相邻的标签。**

CRF与HMM的区别：

1、HMM有两个假设：一个是齐次马尔可夫性假设，二是观测独立性假设。第一个假设表示**一个隐状态的转移概率只与上一个隐状态有关**，第二个假设**限定了隐状态和显状态之间的依赖。**但是在实际情况中，以中文词性标注场景为例，句子中某个字的词性可能不止与上一个字有关，可能与下一个字、上两个字、下两个字有关，所以CRF没有这么严格的独立性假设条件（**CRF假设输出变量之间的联合概率分布构成概率无向图模型**），对于第一个假设CRF有转移特征函数，可以引入全局特征来计算这个转移概率。对于第二个假设，CRF里面的状态特征函数也可以引入全局特征。

2、CRF由HMM的概率图转为函数拟合，可以理解为它的特征函数的输出即是它的一个个特征，我们要学习的东西是特征函数的权重，在预测的时候就和逻辑回归分类器类似（条件随机场的简化形式）。

3、CRF的全局最优在于它是计算完序列概率之后进行归一化（除以规范化因子z(x)）。比如对于一个标注任务，“北京是首都“， 标注为”BIOBI”。

对于HMM的话，其判断这个标注成立的概率为 P= P(B转移到I)*P('北'表现为B)* P(I转移到O)*P('京'表现为I)* ....训练时，要统计状态转移概率矩阵和表现矩阵。

对于CRF的话，其判断这个标注成立的概率为 P= F(B转移到I，'北'表现为B，I转移到O，'京表现为O'，...)。F为一个函数，是在全局范围统计归一化的概率。

4、 学习过程不同，HMM在语料上直接统计（监督学习），CRF一般是迭代学习(自定义特征函数或者让模型自主学习)。

5、HMM是基于概率有向图模型的，CRF是基于概率无向图模型的。

6、HMM是生成式模型，它的发射概率和转移概率由统计得到，对转移概率和表现概率直接建模，统计特征和标签的共现概率。CRF是判别式模型。

平时用的最多的是线性链CRF，线性链的结构如下所示：

一般地，当X和Y具有相同图结构时，线性链结构就变为如下所示：

在上图中，X就为观测序列，Y就为状态序列。同时在给定随机变量序列X的条件下，如果随机变量序列Y相对于序列X的条件概率分布P(YIX)构成条件随机场，那么可得随机变量Y也满足马尔可夫性。公式表达为：

条件随机场的参数化形式

定理 11.2（线性链条件随机场的参数化形式） 设 $P(Y|X)$ 为线性链条件随机场，则在随机变量 X 取值为 x 的条件下，随机变量 Y 取值为 y 的条件概率具有如下形式:

$$P(y|x) = \frac{1}{Z(x)} \exp\left(\sum_{i,k} \lambda_k t_k(y_{i-1}, y_i, x, i) + \sum_{i,l} \mu_l s_l(y_i, x, i)\right) \tag{11.10}$$

其中，

$$Z(x) = \sum_{y} \exp\left(\sum_{i,k} \lambda_k t_k(y_{i-1}, y_i, x, i) + \sum_{i,l} \mu_l s_l(y_i, x, i)\right) \tag{11.11}$$

式中，t_k 和 s_l 是特征函数，λ_k 和 μ_l 是对应的权值。$Z(x)$ 是规范化因子，求和是在所有可能的输出序列上进行的。

式 (11.10) 和式 (11.11) 是线性链条件随机场模型的基本形式，表示给定输入序列 x，对输出序列 y 预测的条件概率。式 (11.10) 和式 (11.11) 中，t_k 是定义在边上的特征函数，称为转移特征，依赖于当前和前一个位置；s_l 是定义在结点上的特征函数，称为状态特征，依赖于当前位置。t_k 和 s_l 都依赖于位置，是局部特征函数。通常，特征函数 t_k 和 s_l 取值为 1 或 0；当满足特征条件时取值为 1，否则为 0。条件随机场完全由特征函数 t_k，s_l 和对应的权值 λ_k，μ_l 确定。

NER问题就是条件随机场，即给定自然语言序列X，最大概率的标注序列Y用来表示NER标注结果。

规范化因子、转移特征、状态特征 对理解CRF模型十分关键，规范化因子是每条路径根据 exp()计算该路径的得分，加和得到Z(x)。

转移特征对应状态转移概率，状态特征对应发射概率。

条件随机场的学习算法（拟牛顿法）

对于条件随机模型：

$$P_w(y|x) = \frac{\exp\left(\sum_{i=1}^{n} w_i f_i(x,y)\right)}{\sum_y \exp\left(\sum_{i=1}^{n} w_i f_i(x,y)\right)} \tag{11.49}$$

学习的优化目标函数是

$$\min_{w \in \mathbf{R}^n} f(w) = \sum_x \tilde{P}(x) \log \sum_y \exp\left(\sum_{i=1}^{n} w_i f_i(x,y)\right) - \sum_{x,y} \tilde{P}(x,y) \sum_{i=1}^{n} w_i f_i(x,y) \tag{11.50}$$

其梯度函数是

$$g(w) = \sum_{x,y} \tilde{P}(x) P_w(y|x) f(x,y) - E_{\tilde{P}}(f) \tag{11.51}$$

具体算法流程如下：

算法 11.2（条件随机场模型学习的 BFGS 算法）

输入：特征函数 $f_1, f_2, \cdots, f_n$；经验分布 $\tilde{P}(X,Y)$；

输出：最优参数值 $\hat{w}$；最优模型 $P_{\hat{w}}(y|x)$。

（1）选定初始点 $w^{(0)}$，取 $\boldsymbol{B}_0$ 为正定对称矩阵，置 $k=0$。

（2）计算 $g_k = g(w^{(k)})$。若 $g_k = 0$，则停止计算；否则转 (3)。

（3）由 $B_k p_k = -g_k$ 求出 p_k。

（4）一维搜索：求 λ_k 使得

$$f(w^{(k)} + \lambda_k p_k) = \min_{\lambda \geqslant 0} f(w^{(k)} + \lambda p_k)$$

（5）置 $w^{(k+1)} = w^{(k)} + \lambda_k p_k$。

（6）计算 $g_{k+1} = g(w^{(k+1)})$，若 $g_{k+1} = 0$，则停止计算；否则，按下式求出 B_{k+1}：

$$B_{k+1} = B_k + \frac{y_k y_k^{\mathrm{T}}}{y_k^{\mathrm{T}} \delta_k} - \frac{B_k \delta_k \delta_k^{\mathrm{T}} B_k}{\delta_k^{\mathrm{T}} B_k \delta_k}$$

其中，

$$y_k = g_{k+1} - g_k, \quad \delta_k = w^{(k+1)} - w^{(k)}$$

（7）置 $k = k+1$，转 (3)。 ■

条件随机场的预测算法

给定条件随机场 P(Y I X) 和输入序列(观测序列)求条件概率最大的输出序列(标记序列)y，即对观测序列进行标注，与HMM相同，使用的也是**维特比解码。**

主要代码

采用scikit-learn库实现CRF的命名实体识别。part2中英文差别很小，下面的展示以中文任务代码为例。

定义特征模板，利用词的上下文信息来构造

在特征的构建中主要考虑了字的标识,是否是数字和字周围的特征信息，这里对应上文中所说的条件随机场的输出不仅和输入有关，也和相邻的输出有关。

```
def word2features(sent, i):
    word = sent[i][0]
    #构造特征字典，考虑到整体句子长度比较长，滑动窗口的大小设置的为6
    features = {
        'bias': 1.0,
        'word': word,
        'word.isdigit()': word.isdigit(),
```

```
    }
    #该字的前一个字
    if i > 0:
        word1 = sent[i-1][0]
        words = word1 + word
        features.update({
            '-1:word': word1,
            '-1:words': words,
            '-1:word.isdigit()': word1.isdigit(),
        })
    else:
        #添加开头的标识 BOS(begin of sentence)
        features['BOS'] = True
    #该字的前两个字
    if i > 1:
        word2 = sent[i-2][0]
        word1 = sent[i-1][0]
        words = word1 + word2 + word
        features.update({
            '-2:word': word2,
            '-2:words': words,
            '-3:word.isdigit()': word2.isdigit(),
        })
    #该字的前三个字
    if i > 2:
        word3 = sent[i - 3][0]
        word2 = sent[i - 2][0]
        word1 = sent[i - 1][0]
        words = word1 + word2 + word3 + word
        features.update({
            '-3:word': word3,
            '-3:words': words,
            '-3:word.isdigit()': word3.isdigit(),
        })
    #该字的后一个字
    if i < len(sent)-1:
        word1 = sent[i+1][0]
        words = word1 + word
        features.update({
            '+1:word': word1,
            '+1:words': words,
            '+1:word.isdigit()': word1.isdigit(),
        })
    else:
    #若改字为句子的结尾添加对应的标识end of sentence
        features['EOS'] = True
    #该字的后两个字
    if i < len(sent)-2:
        word2 = sent[i + 2][0]
        word1 = sent[i + 1][0]
        words = word + word1 + word2
        features.update({
            '+2:word': word2,
            '+2:words': words,
            '+2:word.isdigit()': word2.isdigit(),
        })
    #该字的后三个字
    if i < len(sent)-3:
```

```
        word3 = sent[i + 3][0]
        word2 = sent[i + 2][0]
        word1 = sent[i + 1][0]
        words = word + word1 + word2 + word3
        features.update({
            '+3:word': word3,
            '+3:words': words,
            '+3:word.isdigit()': word3.isdigit(),
        })
    return features
```

数据导入函数

在验证模型效果的时候发现我们给出的中英文验证集格式上有一点小差别：中文验证集最后是两个换行符，英文的是一个换行符，所以中英文数据在读取时最后一行处理要区别对待，但这并非重点，下面展示的是中文数据读取的函数：

```
# 中文数据集读取函数
def load_data(data_path):
    data_read_all = list()
    data_sent_with_label = list()
    with open(data_path, mode='r', encoding="utf-8") as f:
        for line in f:
            if line.strip() == "":
                data_read_all.append(data_sent_with_label.copy())
                data_sent_with_label.clear()
            else:
                data_sent_with_label.append(list(line.strip().split(" ")))
    return data_read_all
```

从数据中提取特征（根据前面定义的特征模板）

```
#从数据中提取特征
def sent2features(sent):
    return [word2features(sent, i) for i in range(len(sent))]

def sent2labels(sent):
    return [ele[-1] for ele in sent]
```

主函数部分

读取数据和特征提取

```
if __name__=='__main__':
    # 读取数据
    train=load_data('./Project2/NER/Chinese/train.txt')
    valid=load_data('./Project2/NER/Chinese/validation.txt')
    print(train)
    print('训练集规模:',len(train))
    print('验证集规模:',len(valid))
    sample_text=''.join([c[0] for c in train[0]])
    sample_tags=[c[1] for c in train[0]]

    X_train = [sent2features(s) for s in train]
    y_train = [sent2labels(s) for s in train]
    X_dev = [sent2features(s) for s in valid]
```

```
    y_dev = [sent2labels(s) for s in valid]
```

训练模型+保存模型：

```
# 训练模型
    crf_model =
sklearn_crfsuite.CRF(algorithm='lbfgs',c1=0.25,c2=0.018,max_iterations=300,
                                    all_possible_transitions=True,verbose=True)
    crf_model.fit(X_train, y_train)

    # 保存模型
    import joblib
    joblib.dump(crf_model, "crf_model2.joblib")
    print(y_dev)
    print(y_pred)
```

实验结果

中文：`f1-score` 达到0.9479

```
              precision    recall  f1-score   support

   micro avg     0.9421    0.9538    0.9479      8437
   macro avg     0.7318    0.7625    0.7446      8437
weighted avg     0.9426    0.9538    0.9480      8437
```

英文：f1-score达到0.8313

```
              precision    recall  f1-score   support

   micro avg     0.9314    0.7507    0.8313      8603
   macro avg     0.9246    0.7410    0.8217      8603
weighted avg     0.9310    0.7507    0.8305      8603
```

完整代码

```
import sklearn_crfsuite
from sklearn_crfsuite import metrics
import warnings
warnings.filterwarnings('ignore')

# 定义通用函数
def load_data(data_path):
    data_read_all = list()
    data_sent_with_label = list()
    with open(data_path, mode='r', encoding="utf-8") as f:
        for line in f:
            if line.strip() == "":
                data_read_all.append(data_sent_with_label.copy())
                data_sent_with_label.clear()
```

```
            else:
                data_sent_with_label.append(list(line.strip().split(" ")))
    return data_read_all
# 定义特征函数
def word2features(sent, i):
    word = sent[i][0]
    features = {
        'bias': 1.0,
        'word': word,
        'word.isdigit()': word.isdigit(),
    }
    #该字的前一个字
    if i > 0:
        word1 = sent[i-1][0]
        words = word1 + word
        features.update({
            '-1:word': word1,
            '-1:words': words,
            '-1:word.isdigit()': word1.isdigit(),
        })
    else:
        #添加开头的标识 BOS(begin of sentence)
        features['BOS'] = True
    #该字的前两个字
    if i > 1:
        word2 = sent[i-2][0]
        word1 = sent[i-1][0]
        words = word1 + word2 + word
        features.update({
            '-2:word': word2,
            '-2:words': words,
            '-3:word.isdigit()': word2.isdigit(),
        })
    #该字的前三个字
    if i > 2:
        word3 = sent[i - 3][0]
        word2 = sent[i - 2][0]
        word1 = sent[i - 1][0]
        words = word1 + word2 + word3 + word
        features.update({
            '-3:word': word3,
            '-3:words': words,
            '-3:word.isdigit()': word3.isdigit(),
        })
    #该字的后一个字
    if i < len(sent)-1:
        word1 = sent[i+1][0]
        words = word1 + word
        features.update({
            '+1:word': word1,
            '+1:words': words,
            '+1:word.isdigit()': word1.isdigit(),
        })
    else:
    #若改字为句子的结尾添加对应的标识end of sentence
        features['EOS'] = True
    #该字的后两个字
    if i < len(sent)-2:
```

```
        word2 = sent[i + 2][0]
        word1 = sent[i + 1][0]
        words = word + word1 + word2
        features.update({
            '+2:word': word2,
            '+2:words': words,
            '+2:word.isdigit()': word2.isdigit(),
        })
    #该字的后三个字
    if i < len(sent)-3:
        word3 = sent[i + 3][0]
        word2 = sent[i + 2][0]
        word1 = sent[i + 1][0]
        words = word + word1 + word2 + word3
        features.update({
            '+3:word': word3,
            '+3:words': words,
            '+3:word.isdigit()': word3.isdigit(),
        })
    return features
#从数据中提取特征
def sent2features(sent):
    return [word2features(sent, i) for i in range(len(sent))]

def sent2labels(sent):
    return [ele[-1] for ele in sent]

if __name__=='__main__':
    # 读取数据
    train=load_data('./Project2/NER/Chinese/train.txt')
    valid=load_data('./Project2/NER/Chinese/validation.txt')
    print(train)
    print('训练集规模:',len(train))
    print('验证集规模:',len(valid))
    sample_text=''.join([c[0] for c in train[0]])
    sample_tags=[c[1] for c in train[0]]

    X_train = [sent2features(s) for s in train]
    y_train = [sent2labels(s) for s in train]
    X_dev = [sent2features(s) for s in valid]
    y_dev = [sent2labels(s) for s in valid]
    print("-----------------------------------分割线-------------------------------
-------")

    # 训练模型
    crf_model =
sklearn_crfsuite.CRF(algorithm='lbfgs',c1=0.25,c2=0.018,max_iterations=300,
                                    all_possible_transitions=True,verbose=True)
    crf_model.fit(X_train, y_train)

    # 保存模型
    import joblib
    joblib.dump(crf_model, "crf_model2.joblib")
    print(y_dev)
    print(y_pred)
```