高级机器学习
大作业

周韧哲 181220076

2021 年 1 月 21 日

学术诚信

本课程非常重视学术诚信规范，助教老师和助教同学将不遗余力地维护作业中的学术诚信规范的建立。希望所有选课学生能够对此予以重视。[1]

(1) 允许同学之间的相互讨论，但是**署你名字的工作必须由你完成**，不允许直接照搬任何已有的材料，必须独立完成作业的书写过程;

(2) 在完成作业过程中，对他人工作（出版物、互联网资料）中文本的直接照搬（包括原文的直接复制粘贴及语句的简单修改等）都将视为剽窃，剽窃者成绩将被取消。**对于完成作业中有关键作用的公开资料，应予以明显引用**;

(3) 如果发现作业之间高度相似将被判定为互相抄袭行为，**抄袭和被抄袭双方的成绩都将被取消**。因此请主动防止自己的作业被他人抄袭。

作业提交注意事项

(1) 请在 LaTeX 模板中**第一页填写个人的姓名、学号信息**;

(2) 本次作业需提交该 pdf 文件、直接可以运行的源码，将以上几个文件压缩成 zip 文件后上传。zip 文件格式为**学号.zip**，例如 170000001.zip；pdf 文件格式为**学号 _ 姓名.pdf**，例如 170000001_ 张三.pdf。

(3) 未按照要求提交作业，或提交作业格式不正确，将会**被扣除部分作业分数**;

(4) 本次作业提交截止时间为**1 月 8 日 23:59:59**。除非有特殊情况（如因病缓交），否则截止时间后不接收作业，本次作业记零分。

[1] 参考尹一通老师高级算法课程中对学术诚信的说明。

1 Introduction

本次的作业为使用条件随机场（conditional random field，CRF）解决 OCR（optical character recognition）问题。

在 CRF 模型中，有两种变量：我们要建模的隐藏变量和始终观察到的变量。对于 OCR，我们要在观察的字符图像（也就是每个图像对应的像素数组）的情况下，对字符（例如 “a” 或 “c”）进行建模。通常来说，未观察到的变量用 Y 表示，观察到的变量用 X 表示。CRF 试图对 $P(Y|X)$ 建模，即给定观察到的图像上字符的条件分布。该模型的结构如1下所示：

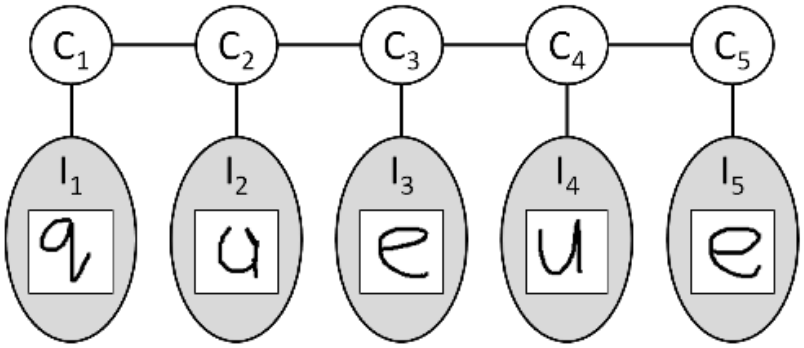

图 1: Markov Network

在 CRF 中，每个特征都对应一个权重 θ_i，在给定特征和权重的情况下，条件概率分布可以表示为：

$$P(\mathbf{Y} \mid \mathbf{x} : \theta) = \frac{1}{Z_{\mathbf{x}}(\theta)} \exp\left\{\sum_{i=1}^{k} \theta_i f_i\left(\mathbf{Y}, \mathbf{x}\right)\right\} \tag{1.1}$$

其中，$Z_x(\theta)$ 为配方函数

$$Z_{\mathbf{x}}(\theta) \equiv \sum_{\mathbf{Y}} \exp\left\{\sum_{i=1}^{k} \theta_i f_i\left(\mathbf{Y}, \mathbf{x}\right)\right\} \tag{1.2}$$

在这次的任务中，一共有三类特征, 三类特征均为指示函数，即满足条件时 $f = 1$, 不满足时 $f = 0$：

- $f^{C}_{i,c}\left(Y_i\right)$，指示是否 $Y_i = c$
- $f^{I}_{i,j,c,d}\left(Y_i, x_{ij}\right)$，指示是否 $Y_i = c, x_{ij} = d$
- $f^{P}_{i,c,d}\left(Y_i, Y_{i+1}\right)$，指示是否 $Y_i = c, Y_{i+1} = d$

建立好模型，给定训练样本，我们就可以使用最大似然估计来进行学习：

$$LL(\mathbf{x}, \mathbf{Y}, \theta) = \sum_{i=1}^{k} \theta_i f_i(\mathbf{Y}, \mathbf{x}) - \log\left(Z_{\mathbf{x}}(\theta)\right) \tag{1.3}$$

对于这个目标，我们可以使用梯度上升算法学习参数。

2 Dataset

本题中的数据集一共包含两个部分 `trainset` 和 `testset`, 分别是训练集和测试集. 训练集中有 400 个样本, 测试集中有 200 个样本. 每个样本被存储在一个 `txt` 文件中, 第一行为对应的单词, 之后的每行为单词的每个字母对应的像素的状态.

3 Assignment

1. 建立 CRF 模型，在训练集上进行训练，使用梯度上升的方法对模型参数进行求解，即求解公式(1.3)(注：不允许使用现有的 CRF 包，使用 python 实现)。
2. 在模型训练完成后，在测试集上进行推断，评价模型的性能。
3. 使用一些其他方法提高模型性能，可参考以下几个方面但不限于此：
   - 提高模型表达能力: 如在 CRF 图上添加新的连接。
   - 缓解模型过拟合: 如添加正则项。
   - 加速模型训练过程：如权重共享。
4. 完成实验报告，主要包含具体的实现方法，如何复现运行代码，对模型的改进以及结果的分析等部分。

4 实验报告

4.1 项目运行

本 repo 的目录树如下：

```
HW3
├── model
│   ├── graph.py
│   └── linear_chain_crf.py
├── utils
│   ├── trainer.py
│   └── utils.py
└── train_crf.py
```

模块 graph 为图模型的基类实现，模块 linear_chain_crf 中为线性链条件随机场的实现，继承了 Graph；trainer 为训练模块的集成，将模型的实现与训练解耦；模块 utils 里为数据的加载及其处理；train_crf 为代码运行的 script，集成了加载、训练与测试的过程。train_crf.py 中可选命令行参数有种子 seed，学习率 lr，批数据大小 batch_size，训练轮数 epoch，正则项系数 lamda，特征中共享变量的特征数 shared_nums 等等。在本 repo 的根目录下命令行键入 python train_crf.py 会按默认参数运行，会得到一个在测试集上正确率为 0.998 的结果。下面我将会介绍我的具体实现与模型改进和对比的结果。

4.2 具体实现

数据的观察序列的维度为 321 维, 状态一共有 10 种。对于题目所给的三类特征 C、I、P, 实际上可以分为状态特征 C 和 I 与转移特征 P。因为状态特征只和当前时间有关，转移特征和当前时间以及下一个时间有关，这样的考虑可以减少变量，简化实现。

4.2.1 加载数据

数据加载实现在 utils/utils.py 中，load_data 函数会直接将原来的 320 维像素（不考虑第一维是因为它全为 1）作为 CRF 的特征，即上面提到的特征 I，而 load_data_c 则会加上特征 C，即考虑每个时间位置的标签类型，因而要比原特征多 10 个维度，实验显示加入这个特征可以提升泛化性能。

4.2.2 CRF 模型

在 model/graph.py 中我定义了一个抽象类 Graph，在其中初始化了特征参数 thetaP 和 thetaI，并定义了一系列方法。LinearChainCRF 继承了类 Graph。实现 CRF 需要计算配分函数 Z，完全遍历的计算会导致组合爆炸问题，因此我使用了信念传播算法来通过传递消息来快速计算边际概率，对应 compute_message 函数。链式 CRF 的消息传递分为前向和后向，即从序列开头和结尾开始传递的消息。首先初始化消息为 1，然后从左到右计算每一个团之间的消息，在当前时刻，它需要向右传递的消息是它接收到的消息乘以此时刻的势函数。在实现中我将势函数按特征分开，在基类中的两个 property potP 和 potI 就是对应特征的势函数。compute_fweight 函数是计算特征和权重的乘积和，即 $\sum_k^K \theta_k f_k$。如此便可以计算出对数似然 $\sum_{k=1}^K \theta_k f_k(\mathbf{Y}, \mathbf{x}) - \log\left(Z_{\mathbf{x}}(\theta)\right)$，对应于 compute_ll 函数。

完成消息传递后就能够计算每个位置的边际分布 $P(Y_i|X)$ 和成对分布 $P(Y_i, Y_{i+1}|X)$，分别对应 compute_marginal_dist 和 compute_pair_dist。从而可以在该 CRF 模型上进行推理：从初始时刻开始，第一个位置被预测为 $\arg\max_i P_1(Y_i|X)$，由于我们知道了接下来时刻位置的边际分布和成对分布，在第 t-1 个时刻位置被预测为 y_{t-1} 的情况下，第 t 个时刻位置将会被预测为 $\arg\max_i P_{t-1}(y_{t-1}, Y_i|X)P_t(Y_i|X)$。

令序列长度为 K, 特征数记为 F, 标签数记为 M, 给定了参数 thetaI($M \times F$) 和 thetaP($M \times M$) 下，为了计算导数，首先将其对数似然函数展开：

$$LL(X, Y, \theta_I, \theta_P) = \sum_{i=1}^{K}\sum_{f=1}^{F}\theta^I_{y_i,f}X_{if} + \sum_{i=1}^{K-1}\theta^P_{y_i y_{i+1}} \times 1 - \log Z$$

对 thetaP 求导得

$$\begin{aligned}
\frac{\partial LL}{\partial \theta^P_{y',y_}} &= \sum_{i=1}^{K-1}\frac{\partial \theta^P_{y_i,y_{i+1}}}{\partial \theta^P_{y',y_}} - \frac{1}{Z}\frac{\partial}{\partial \theta^I_{y',y_}}\sum_y \exp(\sum_{i=1}^{K}\sum_{f=1}^{F}\theta^I_{y_i,f}X_{if} + \sum_{i=1}^{K-1}\theta^P_{y_i y_{i+1}} \times 1) \\
&= \sum_{i=1}^{K-1}\mathbb{I}[y_i = y', y_{i+1} = y_] - \frac{1}{Z}\frac{\partial}{\partial \theta^I_{y',y_}}\sum_y \exp(\sum_{i=1}^{K-1}\theta^P_{y_i y_{i+1}} \times 1)\mathbb{I}[y_i = y', y_{i+1} = y_] \\
&= \sum_{i=1}^{K-1}\mathbb{I}[y_i = y', y_{i+1} = y_] - \sum_y P(y|X)\mathbb{I}[y_i = y', y_{i+1} = y_]
\end{aligned}$$

表 1: 权重共享 (单位: 秒)，CPU：Intel Core i5-9300HF

shared_nums	1	4	8	16	20
mean	14.05	12.41	11.92	11.31	11.45
std	0.68	0.24	0.36	0.41	0.39

对 thetaI 求导得

$$\begin{aligned}
\frac{\partial LL}{\partial \theta^I_{y',f'}} &= \sum_{i=1}^{K}\sum_{f=1}^{F}\frac{\partial \theta^I_{y_i,f}}{\partial \theta^I_{y',f'}}X_{if} - \frac{1}{Z}\frac{\partial}{\partial \theta^I_{y',f'}}\sum_y \exp(\sum_{i=1}^{K}\sum_{f=1}^{F}\theta^I_{y_i,f}X_{if} + \sum_{i=1}^{K-1}\theta^P_{y_iy_{i+1}} \times 1) \\
&= \sum_{i=1}^{K}\sum_{f=1}^{F}\mathbb{I}[y_i = y', f = f']X_{if} - \frac{1}{Z}\frac{\partial}{\partial \theta^I_{y',f'}}\sum_y \exp(\sum_{i=1}^{K}\sum_{f=1}^{F}\theta^I_{y_i,f}X_{if} + \sum_{i=1}^{K-1}\theta^P_{y_iy_{i+1}} \times 1) \\
&= \sum_{i=1}^{K}\sum_{f=1}^{F}\mathbb{I}[y_i = y', f = f']X_{if} - \frac{1}{Z}\frac{\partial}{\partial \theta^I_{y',f'}}\sum_y \exp(\sum_{i=1}^{K}\sum_{f=1}^{F}\theta^I_{y_i,f}X_{if} + \sum_{i=1}^{K-1}\theta^P_{y_iy_{i+1}} \times 1) \\
&\qquad \times \mathbb{I}[y_i = y', f = f']X_{if} \\
&= \sum_{i=1}^{K}\sum_{f=1}^{F}\mathbb{I}[y_i = y', f = f']X_{if} - \sum_y P(y|X)\mathbb{I}[y_i = y', f = f']X_{if}
\end{aligned}$$

详见 commpute_grad 函数。在实现中我也加入了 L2 正则项 $\frac{\lambda}{2}\|\theta\|_2^2$，其导数就是 $\lambda\theta$，因此只需让上面的梯度加上 $\lambda\theta$ 即可。

4.2.3 训练与测试

在 trainer 模块中实现了一个用于管理训练过程的类 Trainer，其函数 train 实现了批量梯度上升，在每个 epoch 中，先把训练集打乱，然后计算 batch_size 个训练样本对的平均梯度用于梯度上升。每完成一个 epoch 后，就会计算当前的目标函数值，即对数似然值，并在测试集上推断，打印出 accuracy。训练完成后，会打印出运行时间。

4.3 实验结果与总结

我做了以下几个实验来探讨模型的不同改进点的性能差异。

4.3.1 权重共享

不同的权重共享对模型的影响。对于图片的 320 个像素值，设权重共享数为 k，则每 k 个共享一个权值，表1是当 k 为 1,4,8,16,20 时的运行时间表，每个运行了 5 次，epoch 为 100，随机种子固定为 0，学习率为 0.01，batch_size 为 16，正则化系数 λ 为 0，都使用了 C 特征，因而维度要多出 10 维，其对应的权重矩阵维度分别为 330,90,50,30,26。由表可知，这几种情况的运行时间相差不大，最长的仅比最短的多 3 秒左右，而共享权重数越大时，性能越不好，见图2，横轴为迭代次数，纵轴分别为 Accuracy 和 LogLikeliHood。当 k 为 1 和 4 时准确率都达到了 0.99，且它们在较短的迭代次数内就到达了一个较高的准确率。

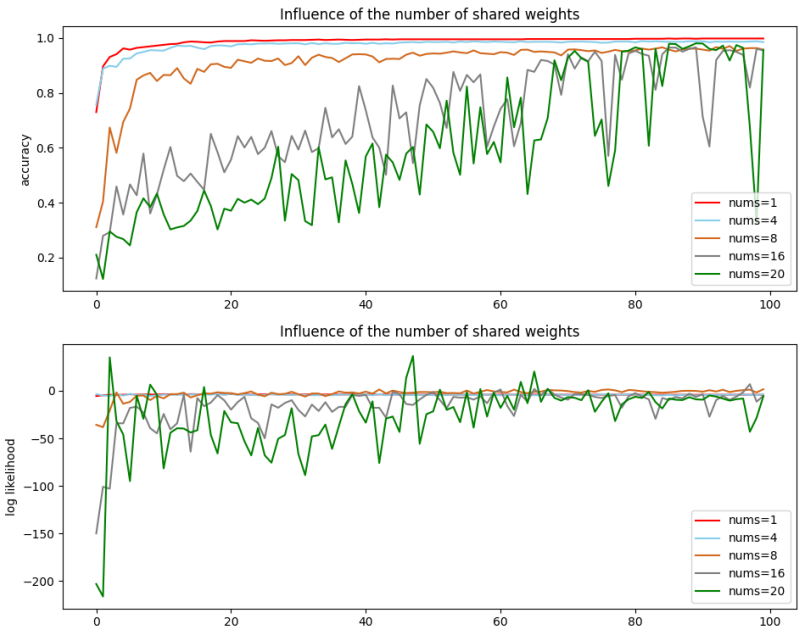

图 2: Shared nums

4.3.2 正则化

我测试了 5 种不同的正则化系数 λ 为 0,0.005,0.05,0.5,5，权重共享数为 1, 其他实验设置与上一小节一致，见图3，横轴为迭代次数，纵轴分别为 Accuracy 和 LogLikeliHood，发现正则化系数较小时并没有太大的变化，而较大时会导致模型欠拟合，效果不佳。我认为是因为这个任务的特征提取得比较好，模型比较容易学，因而正则化不会导致太大的作用，并且在没有正则化时，模型的泛化性能已经很好了。

4.3.3 使用特征 C

最后我探讨了使用特征 C 与否造成的影响，设置正则化系数为 0, 其他实验设置与上一小节一致，在 train_crf.py 中可以通过命令行参数 not_use_c 来控制是否使用特征 C。由图4容易看出，加入特征 C 提高了模型在测试集上的最高准确率，对泛化性能有较不错的提升。因为特征 C 指示了当前处于哪种状态，而在英语中不同字母在单词的不同位置出现的概率是不一样的，将这一特征加入可以让模型学得更好。

4.3.4 实验总结

用条件随机场做此任务十分合适，因为单词是不定长序列，且是有时序位置关系的，因而模型能取得很好的效果。难点在于构建特征和消息传播算法，如果单词长度短的话，直接遍历所有可能序列是在时间上可以接受的，但这次任务的数据集中存在 8、9 个字母的单词，遍历会导致时间复杂度很高，运行时间很长。

最后，祝助教学长和老师新年快乐!

图 3: Regulariation

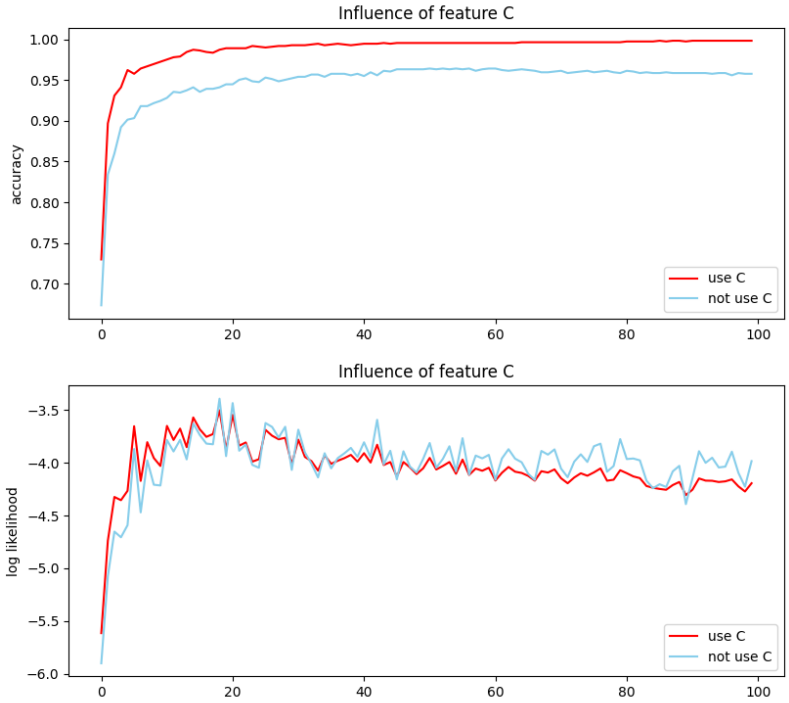

图 4: 特征 C