总结

黄飞虎

2021.2.24

学习情况

完成了迁移学习的实验

特征提取器

```python
class FeatureExtractor(nn.Module):

    def __init__(self):
        super(FeatureExtractor, self).__init__()

        self.conv = nn.Sequential(
            nn.Conv2d(1, 64, 3, 1, 1),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.MaxPool2d(2),

            nn.Conv2d(64, 128, 3, 1, 1),
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.MaxPool2d(2),

            nn.Conv2d(128, 256, 3, 1, 1),
            nn.BatchNorm2d(256),
            nn.ReLU(),
            nn.MaxPool2d(2),

            nn.Conv2d(256, 256, 3, 1, 1),
            nn.BatchNorm2d(256),
            nn.ReLU(),
            nn.MaxPool2d(2),

            nn.Conv2d(256, 512, 3, 1, 1),
            nn.BatchNorm2d(512),
            nn.ReLU(),
            nn.MaxPool2d(2)
        )

    def forward(self, x):
        x = self.conv(x).squeeze()
        return x
```

满足

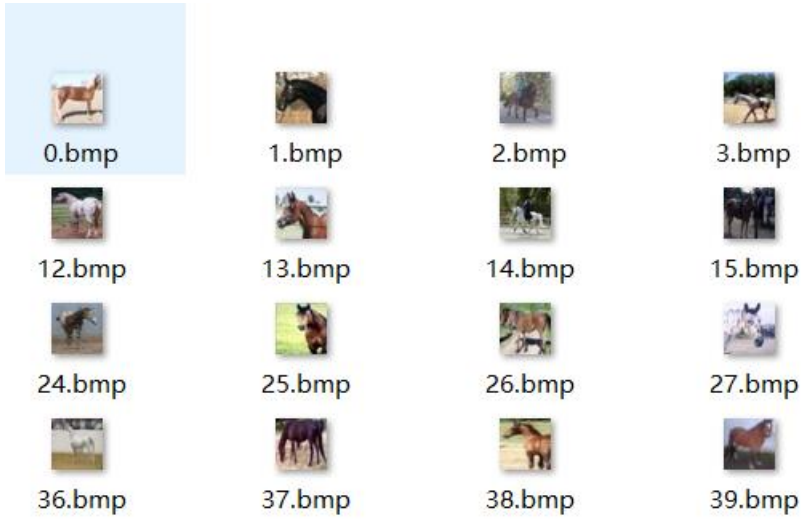
骗过

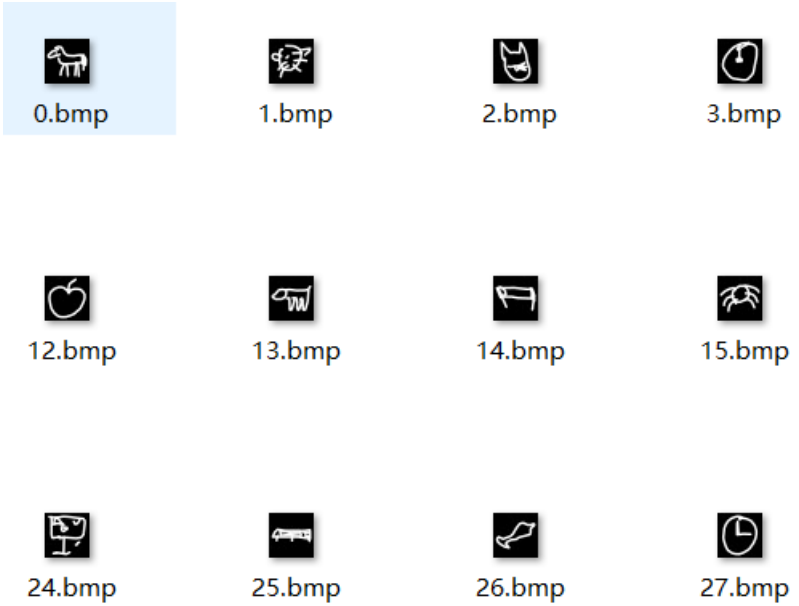
标签预测器

```python
class LabelPredictor(nn.Module):

    def __init__(self):
        super(LabelPredictor, self).__init__()

        self.layer = nn.Sequential(
            nn.Linear(512, 512),
            nn.ReLU(),

            nn.Linear(512, 512),
            nn.ReLU(),

            nn.Linear(512, 10),
        )

    def forward(self, h):
        c = self.layer(h)
        return c
```

域分类器

```python
class DomainClassifier(nn.Module):

    def __init__(self):
        super(DomainClassifier, self).__init__()

        self.layer = nn.Sequential(
            # y = x * W^T + b
            nn.Linear(512, 512),  # 这里（512, 512）是W的维度，
            nn.BatchNorm1d(512),
            nn.ReLU(),

            nn.Linear(512, 512),
            nn.BatchNorm1d(512),
            nn.ReLU(),

            nn.Linear(512, 512),
            nn.BatchNorm1d(512),
            nn.ReLU(),

            nn.Linear(512, 512),
            nn.BatchNorm1d(512),
            nn.ReLU(),

            nn.Linear(512, 1),
        )

    def forward(self, h):
        y = self.layer(h)
        return y
```






```
id,label
0,0
1,7
2,3
3,3
4,3
5,9
6,1
7,4
8,8
9,1
10,1
11,3
12,3
13,1
14,1
15,5
16,4
17,1
18,0
19,8
20,7
21,3
22,8
23,1
24,5
25,2
26,8
```

```
epoch 185: train D loss: 0.3758, train F loss: 0.0162, acc 0.9838
epoch 186: train D loss: 0.3747, train F loss: 0.0201, acc 0.9820
epoch 187: train D loss: 0.3654, train F loss: 0.0128, acc 0.9828
epoch 188: train D loss: 0.3729, train F loss: 0.0296, acc 0.9782
epoch 189: train D loss: 0.3722, train F loss: 0.0197, acc 0.9808
epoch 190: train D loss: 0.3652, train F loss: 0.0144, acc 0.9844
epoch 191: train D loss: 0.3828, train F loss: 0.0159, acc 0.9842
epoch 192: train D loss: 0.3748, train F loss: 0.0175, acc 0.9848
epoch 193: train D loss: 0.3777, train F loss: 0.0380, acc 0.9760
epoch 194: train D loss: 0.3839, train F loss: 0.0247, acc 0.9788
epoch 195: train D loss: 0.3603, train F loss: 0.0280, acc 0.9808
epoch 196: train D loss: 0.3691, train F loss: 0.0208, acc 0.9804
epoch 197: train D loss: 0.3742, train F loss: 0.0214, acc 0.9812
epoch 198: train D loss: 0.3750, train F loss: 0.0169, acc 0.9838
epoch 199: train D loss: 0.3629, train F loss: 0.0172, acc 0.9828
```

论文情况

A Deep Reinforced Sequence-to-Set Model for Multi-Label Classification

2019年ACL的论文

本文的工作

对于多标签分类问题，以往的模型都是通过最大似然估计方法训练的序列-序列(Seq2Seq)模型，能够捕获标签之间的高阶相关性。但是输出标签还是一个无序集，这会导致一些棘手的问题，比如对标签的敏感性。

提出了一个简单有效的序列到集合的模型，通过强化学习训练，其中的奖励反馈被设计成独立于标签顺序。这样就能减少模型对标签的依赖，并且还是能捕捉标签之间的高阶相关性。

方法

Eecoder

将编码器E实现为一个双向LSTM，给定输出文本（$x_1, x_2, , , x_m$），编码器计算每个词的隐藏状态：

$$\overrightarrow{h}_i = \overrightarrow{\text{LSTM}}(\overrightarrow{h}_{i-1}, e(x_i))$$
$$\overleftarrow{h}_i = \overleftarrow{\text{LSTM}}(\overleftarrow{h}_{i+1}, e(x_i))$$

其中e（x_i）是x_i的嵌入，第i个词的最终表达是 $h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i]$ ，其中分号表示向量级联。

Set Decoder

由于LSTM具有强大的建模序列依赖性的能力，因此也将D作为LSTM模型来实现，以捕获标签之间的高阶相关性。特别地，在t时刻，集合解码器D的隐藏状态s_t计算为：

$$s_t = \mathrm{LSTM}\big(s_{t-1}, [e(y_{t-1}); c_t]\big)$$

在$[e(y_{t-1}); c_t]$表示向量的级联$e(y_{t-1})$和c_t, $e(y_{t-1})$是标签的嵌入y_{t-1}在上一个时间步生成的，c_t是通过注意机制获得的上下文向量。最后，集合解码器D从输出概率分布中对标签y_t进行采样，计算如下

$$o_t = \mathbf{W}_2 f(\mathbf{W}_1 s_t + \mathbf{U} c_t)$$
$$y_t \sim \mathrm{softmax}(o_t + I_t)$$

其中W_1、W_2、U为可训练参数，f为非线性激活函数，其I_t是为防止D产生重复标签的掩码向量。

模型训练

为了减轻模型对标签顺序的依赖，这里我们将多标签分类任务建模为一个RL问题。我们的集合解码器D可以看作是一个代理，它在t时刻的状态是当前生成的标签($y_1,\cdots, y_{t-1}$)。由参数D定义的随机策略决定动作，即对下一个标签的预测。一旦生成完整的标签序列y, 代理D将得到奖励r。训练目标是最小化负的期望奖励，具体如下:

$$\mathcal{L}(\theta) = -\mathbb{E}_{\boldsymbol{y}\sim p_\theta}[r(\boldsymbol{y})]$$

在我们的模型中，我们使用了自批判策略梯度算法。对于minibatch中的每个训练样本，Eq.(6)的梯度近似为

$$\nabla_\theta \mathcal{L}(\theta) \approx -\left[r(\boldsymbol{y}^s) - r(\boldsymbol{y}^g)\right] \nabla_\theta \log\left(p_\theta(\boldsymbol{y}^s)\right)$$

其中y^s为概率分布p采样的标签序列，y^g为贪婪搜索算法生成的标签序列

将生成的标签y与ground-truth标签y*进行比较，设计出F1的积分作为奖励r

$$r(\boldsymbol{y}) = \mathrm{F}_1(\boldsymbol{y}, \boldsymbol{y}^*)$$

实验

数据集：在RCV1-V2语料库上进行实验

Baseline：

BR-LR：相当于为每个标签独立训练一个二元分类器(逻辑回归)。

PCC-LR：将MLC任务转换为二进制分类(逻辑回归)问题链。

FastXML：学习训练实例的层次结构，并在层次结构的每个节点上优化目标。

XML-CNN：使用动态最大池机制和隐藏的瓶颈层来更好地表示文档。

CNN-RNN：提出了一种CNN和RNN的集成方法来捕获全局和局部文本语义。

Seq2Seq：采用Seq2Seq模型进行多标签分类

评价指标：计算误分率的子集0-1损失，表示误预测标签占总标签的比例的汉明损失，以及表示每个类的F1分的加权平均值的micro-F1。

实验结果

Models	HL (–)	0/1 Loss (–)	F1 (+)	Precision (+)	Recall (+)
BR-LR (Boutell et al., 2004)	0.0083	0.393	0.858	0.919	0.804
PCC-LR (Read et al., 2011)	0.0079	0.325	0.864	0.901	0.827
FastXML (Prabhu and Varma, 2014)	0.0078	0.358	0.863	**0.956**	0.786
XML-CNN (Liu et al., 2017)	0.0086	0.390	0.853	0.914	0.799
CNN-RNN (Chen et al., 2017)	0.0085	0.378	0.856	0.889	0.825
Seq2Seq (Yang et al., 2018)	0.0076	0.332	0.871	0.906	0.838
Seq2Set (Ours)	**0.0073**	**0.314**	**0.879**	0.900	**0.858**

Table 1: Performance of different systems. "**HL**", "**0/1 Loss**", "**F1**", "**Precision**", and "**Recall**" denote hamming loss, subset zero-one loss, micro-F_1, micro-precision, and micro-recall, respectively. "**+**" indicates higher is better and "**–**" is opposite. The best performance is highlighted in bold.

为了验证可以降低标签顺序的敏感性，随机打乱标签序列的顺序，下图展示了不同模型在标签变换的RCV1-V2数据集上的性能。

Models	**HL (–)**	**0/1 Loss (–)**	**F1 (+)**
BR	0.0083 (↓0.0%)	0.393 (↓0.0%)	0.858 (↓0.0%)
Seq2Seq	0.0083 (↓9.2%)	0.363 (↓9.3%)	0.859 (↓1.4%)
Seq2Set	0.0075 (↓2.7%)	0.318 (↓1.2%)	0.876 (↓0.3%)

Table 2: Comparison on the label-shuffled RCV1-V2 dataset. "↓" indicates that the model is degraded.

Thank you