Week3

1. 要求：

```
多分类任务：要求得到字符串中字符的位置，如判断a在“bsadcd”中的位置，应该输出为第3类
● 提示：用RNN实现
● 缩短字符集长度，否则可能a出现的概率很小
```

2. 解决方案

a. 构建数据集

我们选择生成数据集中，每个样本有且仅有一个'a'，并让模型判断'a'所处在的位置。因此，若想生成长度为6的字符串，则可以从词表中删去'a'，并随机可重复地选取5个字符，最后再将'a'随机插入到该字符串中。

```python
def build_sample(vocab, sentence_length):
    #随机从字表选取sentence_length个字，可能重复
    chars = list(vocab.keys())
    chars.remove('a')

    x = [random.choice(chars) for _ in range(sentence_length-1)]
    y = random.randint(0,5)
    x.insert(y, 'a')
    x = [vocab.get(word, vocab['unk']) for word in x]   #将字转换成序号，为了做embedding
    return x, y
```

b. 模型构建

由于该任务存在序列关系，用RNN较为合适。实际上RNN的使用与线性层非常类似，只需要考虑inputSize和outputSize，区别在于RNN能够将位置信息嵌入进去。

```python
class TorchModel(nn.Module):
    def __init__(self, vector_dim, sentence_length, vocab):
        super(TorchModel, self).__init__()
        self.embedding = nn.Embedding(len(vocab), vector_dim)  #embedding层
        # self.pool = nn.AvgPool1d(sentence_length)   #池化层
        self.classify = nn.Linear(vector_dim, 1)     #线性层
        self.activation = torch.sigmoid     #sigmoid归一化函数
        self.RNN = nn.RNN(vector_dim, vector_dim, bias=False, batch_first=True)
        self.loss = nn.functional.cross_entropy  #loss函数采用交叉熵损失

    #当输入真实标签，返回loss值；无真实标签，返回预测值
    def forward(self, x, y=None):
        x = self.embedding(x)                           #(batch_size, sen_len) -> (batch_size, sen_len, vector_dim)
        x, h = self.RNN(x)                              #(batch_size, sen_len, vector_dim) ->(batch_size, sen_len, vector_dim)
        x = self.classify(x)                            #(batch_size, sen_len, vector_dim) -> (batch_size, sen_len, 1)
        x = x.squeeze()                                 #(batch_size, sen_len, 1) -> (batch_size, sen_len)
        y_pred = self.activation(x)                     #(batch_size, sen_len) -> (batch_size, sen_len)
        if y is not None:
            return self.loss(y_pred, y)   #预测值和真实值计算损失
        else:
            return y_pred                 #输出预测结果
```

其中激活函数用sigmod，对于预测精度影响不大，但可以快速降低loss