实验二：基于 HMM 的语音识别实验

(语音信号数字处理课程报告)

姓　　名：肖文韬

学　　号：2020214245

二〇二〇年十二月二十日

目 录

插图清单

第 1 章　任务一：Viterbi 解码算法实现

Forward 和 Backward 算法对应于讲义中的 Q1 (Evaluation)，Vertibi 算法对应于讲义中的 Q2 (Decoding)。代码实现可以使用 `sanity_grader_hmm()` 测试是否正确。算法的核心思路及实现如下：

1.1　Forward 算法

输入：

1. O: observations
2. π: initial probability
3. A: hidden state transition matrix
4. B: emission matrix

输出：

$$\begin{aligned} P(O|\lambda) &= \sum_{Q} P(O,Q|\lambda) \\ &= \sum_{Q} P(O|Q,\lambda)P(Q|\lambda) \\ &= \sum_{Q} \prod_{t=1}^{T} B(q_t,o_t)\pi(q_1)\prod_{t=2}^{T} A(q_{t-1},q_t) \end{aligned} \tag{1-1}$$

公式 1-1 可以转换成矩阵运算：

$$\begin{aligned} \pi^{(0)} &= \pi \\ \text{Fwd}(o_1) &= \pi^{(1)} = \pi^{(0)} * B(:,o_1), P(o_1) = sum(\text{Fwd}(o_1)) \\ &\vdots \\ \text{Fwd}(o_1,\cdots,o_T) &= \pi^{(T)} = \pi^{(T-1)} * B(:,o_T) \\ P(o_1,\cdots,o_T) &= sum(\text{Fwd}(o_1.\cdots,o_T)) \end{aligned} \tag{1-2}$$

算法复杂度：$O(T * N^2)$

实现代码：

```
for t, o_t in enumerate(ob):
  fwd[t] = pi_t * B[:, o_t]
  pi_t = fwd[t] @ A
```

1.2　Backward 算法

Backward 算法目标与 Forward 一样，只不过迭代顺序是从后往前。代码：

```
beta_t = np.ones(total_states)
for t, o_t in enumerate(reversed(ob)):
  bwd[-1-t] = beta_t.T
  beta_t = A @ (B[:, o_t] * beta_t)
```

1.3　Viterbi 算法

目标：

启发式搜索 $Q^* = \arg\max_Q P(Q|O,\lambda)$：

$$
\begin{aligned}
Q^* &= \arg\max_Q P(Q|O,\lambda) \\
&= \arg\max_Q P(Q,O|\lambda)/P(O|\lambda) \\
&= \arg\max_Q P(Q,O|\lambda) \\
&= \arg\max_Q P(O|Q,\lambda)P(Q|\lambda) \\
&= \arg\max_Q \prod_{t=1}^{T} B(q_t,o_t)\pi(q_1)\prod_{t=2}^{T} A(q_{t-1},q_t)
\end{aligned}
\tag{1-3}
$$

同样地，类似 Forward 算法，上式 1-3 可以优化为迭代形式：

$$
\delta_t(j) = \arg\max_{q_1,q_2,\cdots,q_{t-1}} P(q_1,\cdots,q_{t-1},q_t=s_j,o_1,\cdots,o_t|\lambda) \tag{1-4}
$$

初始化：

$$\delta_1 = \pi * B(:, o_1) \tag{1-5}$$

递推式：

$$\begin{aligned} \phi_t &= \arg\max(\delta_{t-1} \odot A^T, \text{axis} = 1) \\ \delta_t &= \max(\delta_{t-1} \odot A^T, \text{axis} = 1) * B(:, o_t) \end{aligned} \tag{1-6}$$

代码实现：

```
delta_t = pi
phi_t = np.zeros(total_states)
for t, o_t in enumerate(ob):
  delta_t *= B[:, o_t]
  delta[t] = delta_t
  phi[t] = phi_t
  phi_t = np.argmax(delta_t * A.T, axis=1)
  delta_t = np.max(delta_t * A.T, axis=1)
```

第 2 章　任务二：基于 GMM-HMM 的语音识别

2.1　Q1 (3")

Q: Look at the directory data/train, describe what is contained in files text, wav.scp and utt2spk respectively (Hint: all those files can be seen as key-value dicts).

A:

1. text: 每一行地第一个元素是 utterance-id，可以为任意字符串。后面的部分就是每一句的录音对应的文本（字幕), 如果有词不在字典中（out of vocabulary），将会自动映射到 data/lang/oov.txt 中指定的词。
2. wav.scp: 格式为 `<recording-id> <extended-filename>`, extended-filename 可以为音频文件，也可以是能够返回出 wav 音频文件的任意命令。如果不存在 segments 文件，recording-id 就会自动用作 utterance-id。
3. utt2sp: 每一行的格式为 `<utterance-id> <speaker-id>`，用于标识每一个发音对应的发音者（speaker）。

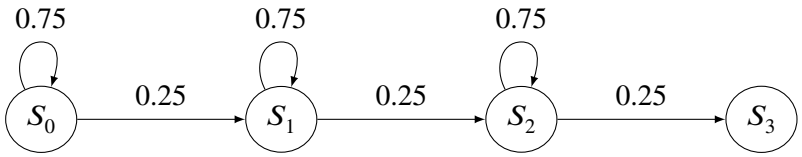
需要注意的有：

1. 这些文件中的顺序需要对应起来
2. wav.scp 中的音频文件必须是单通道的，否则需要用 sox 命令提取出指定的 channel。
3. utterance-id, speaker-id 都推荐各自使用固定长度，否则可能回导致 C-style string order 出问题。

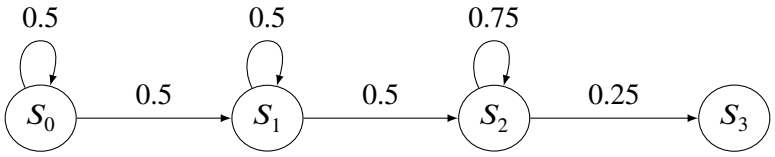
2.2　Q2 (3")

Q: Look at the file data/lang/topo, which contains two kinds of HMM topology, draw them using circles and arrows like this. You may notice that the HMM topology of a special phoneme is different from other phonemes. Use data/lang/phones.txt to map and find the name of the special phoneme.

A: data/lang/topo 中的两个 HMM 拓扑图如图 2.1 所示，第二个拓扑图对应的是 **沉默和噪声音素（silence and noise phonemes）**。通过 topo 文件中的 `ForPhones` 字段（在本例中为 1），我们可以在 phones.txt 中找到对应的音素名称为 `sil`。



(a) 拓扑图 1 (real phonemes)

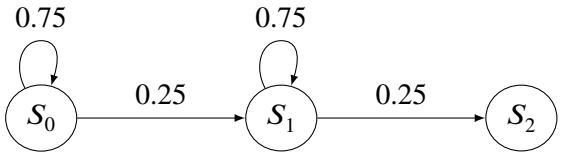


(b) 拓扑图 2 (silence and noise phonemes)

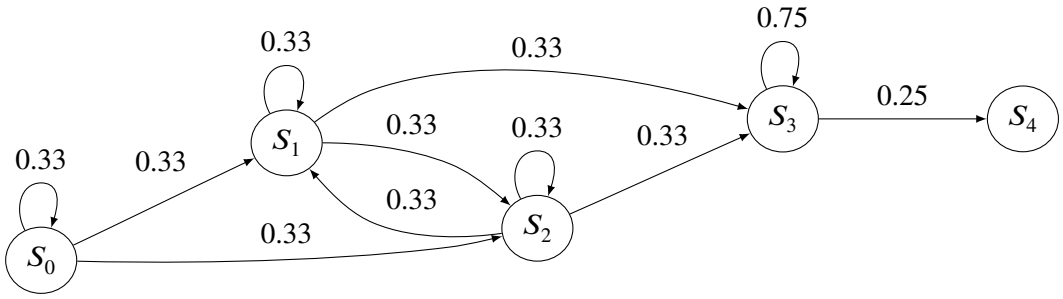
图 2.1　Q2 两种拓扑图

2.3　Q3 (2")

Q: You can change num-sil-states and num-nonsil-states for command utils/prepare_lang.sh in run.sh:line59, then run this stage again and draw new topologies from data/lang/topo. (Note that set them to default values(sil=3 and nonsil=3) and rerun this stage before proceeding to the next stage since other values may affect the performance).



(a) 拓扑图 1 (real phonemes)



(b) 拓扑图 2 (silence and noise phonemes)

图 2.2　Q3 修改参数后的两种拓扑图

A: 将 `num-sil-states` 和 `num-nonsil-states` 分别修改为 2 和 4, 得到的对 real phonemes 和 silence and noise phonemes 的 HMM 的状态就会变为 2 和 4。具体的拓扑图见图 2.2。

2.4　Q4 (7")

Q: Describe the process of calculating MFCC, docs: link.

A: 首先从音频文件中提取出帧（frame），通常是 25ms 一帧，每次滑动 10ms。然后对于每一帧：

1. 提取数据，进行可选的预处理：抖动（dithering），预加权（preemphasis），dc 偏移消除（dc offset removal），最后乘以窗函数（例如 Hamming 窗）。
2. 计算当前帧的能量（如果使用 log-energy 而不是 C0）
3. 做快速傅立叶变换（FFT），计算能量谱（power spectrum）
4. 计算每 mel bin 的能量，例如，23 个重叠矩形 bin，并且这些 bin 的中心点在 mel 频域上是等距的。频率的上下限一般取 Nyquist 频率（例如 7800，对于 16kHz 采样的音频）和靠近 0 的频率（例如 20）。
5. 将这些能量取 log 然后进行余弦变换，保留指定数量（例如 13）的系数。
6. 可选的倒谱提升（cepstral liftering），这就是将这些系数做一个缩放，保证它们在一个合理的范围内。

2.5　Q5 (1")

Q: Run following commands (Note: ark means archive file with binary format, ark,t means archive file with utf-8 text format), check file raw_mfcc_test.1.txt and answer the dimention of MFCC features. (Hint: count the number of columns)

```
source ./path.sh
copy-feats ark:mfcc/raw_mfcc_test.1.ark ark,t:raw_mfcc_test.1.txt
```

A: MFCC 特征共有 13 个。

2.6　Q6 (1")

Q: Look at the directory data/train, describe what is contained in file feats.scp, docs: link.

A: 每一行由 `<key> <rxfilename>:<byte_offset>` 组成，其中 key 表示一个 utterance id, rxfilename 表示数据所在的文件名，byte_offset 表示字节偏移量。

2.7　Q7 (1")

Q: Describe the role of script utils/split_scp.pl in steps/make_mfcc.sh:line133. (Hint: type wc mfcc/raw_mfcc_train.1.scp and wc data/train/feats.scp, check the value of feats_nj in run.sh and compare the outputs of the above commands)

A: 将 data/train/feats.scp 拆分为大致等长的 feats_nj 个 scp 文件。

2.8　Q8 (1")

Q: How many MonoPhones do we use ? (Hint: check exp/mono/phones.txt, the MonoPhones we mean here do not contain eps, #0 and #1)

A: 48

2.9　Q9 (1")

Q: If we choose 3-state HMM to model all those MonoPhones, how many states will we have ? (Hint: run the following commands and count the number of Triples in final.mono.mdl.txt)

```
source ./path.sh
gmm-copy --binary=false exp/mono/final.mdl final.mono.mdl.txt
```

A: 144

2.10　Q10 (2")

Q: According to the number of MonoPhones in the previous question, how many candidates are there for TriPhones and TriPhones' HMM states (suppose each HMM has 3 states) ?

A: 理论上需要 $48 \times 48 \times 48 = 110,592$ 个 TriPhones，并且需要 $110592 \times 3 = 331,776$ 个 HMM 状态。

2.11　Q11 (2")

Q: Refer to Q9, use the correct command to view the TriPhone system (exp/tri1/final.mdl) and count the actual number of HMM states. You may find that the

actual number of HMM states is much smaller than the theoretical value calculated in Q10, can you explain why ? (Hint: PPT5:page38 page42)

A: 使用命令 `gmm-copy --binary=false exp/tri1/final.mdl final.tri1.mdl.txt` 提取，TriPhone 的 HMM 实际上有状态为 1929 个，远小于理论值。这是因为理论值太大，所以我们需要使用决策树来做一些聚类，`run.sh` 中的 `numLeavesTri1` (本例为 2500) 参数用于指定这个决策树的最大叶结点数。

第 3 章　任务三：基于 DNN-HMM 的语音识别

3.1　Q1 (1")

Q: Explain what is force alignment. docs: link

A: 强制对齐（force alignment）是按照说话音频段对应文本，确定每一个出现的词对应与音频段中的具体时间。

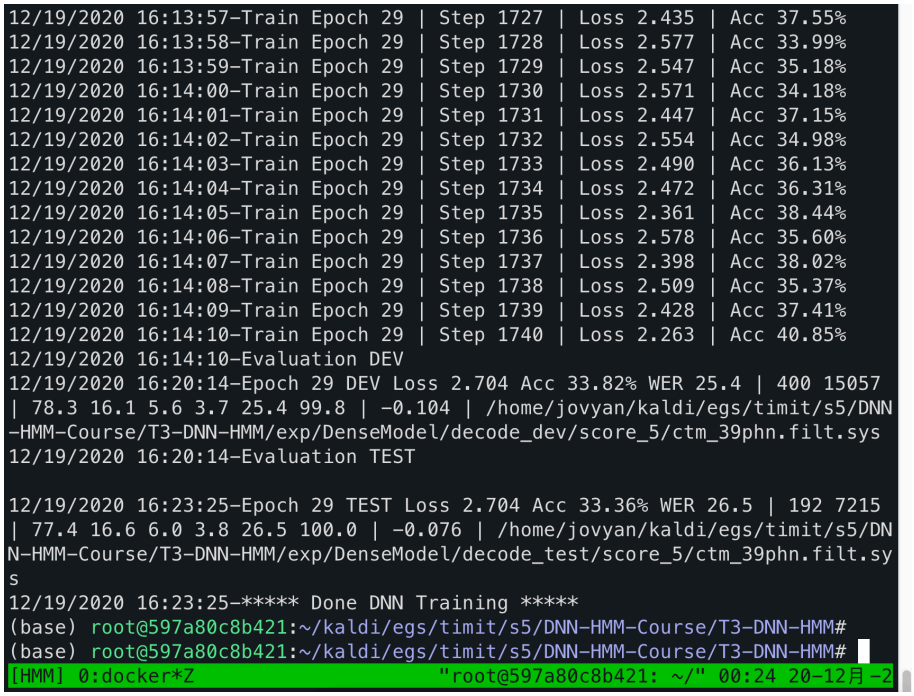


图 3.1　DNN 训练结果

3.2　Q2 (1")

Q: What is the dimention of transformed MFCC features ?

A: 39

3.3　Q3 (2")

Q: Explain why we need to subtract prior in train.py:line114-line115

A: 因为 $P(f1|hh) = P(hh|f1) * P(f1)/P(hh)$，而 DNN 模型只能得到后验概率（$P(hh|f1)$），我们需要除以 prior （先验概率）来得到似然概率（$P(f1|hh)$）。取对数后除法就变成了减法了。

3.4　DNN 实验结果

训练 30 Epochs 后，得到 DEV 验证集结果为 `Acc 33.82% WER 25.4`，Test 测试集结果为 `Acc 33.36% WER 26.5`，如图 3.1 所示。

参考文献