

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

二〇二〇年十二月十四日

目 录

目录.....	I
插图清单.....	II
第 1 章 任务一：Viterbi 解码算法实现.....	1
1.1 Forward 算法.....	1
1.2 Backward 算法.....	2
1.3 Viterbi 算法.....	2
第 2 章 任务二：基于 GMM-HMM 的语音识别.....	4
2.1 Q1 (3").....	4
2.2 Q2 (3").....	4
2.3 Q3 (2").....	5
第 3 章 任务三：基于 DNN-HMM 的语音识别.....	6
参考文献.....	7

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), Viterbi 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：

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

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：

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

初始化：

$$\delta_1 = \pi * B(:, o_1) \quad (1-5)$$

递推式：

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

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. utt2spk: 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:

2.3 Q3 (2")

Q: You can change num-sil-states and num-nonsil-states for command `utils/prepare_lang.sh` in `run.sh`, 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).

第 3 章 任务三：基于 DNN-HMM 的语音识别

参考文献