ip2hw2

c.hw

March 2024

1 孤立词识别器项目

这个项目的目标是为 48 个词的词汇构建一个孤立词识别器。

- 语音信号已经被处理, 从固定帧率 (100 帧/秒) 的小窗口 (25 毫秒) 中提取频谱特征, 得到的特征向量已量化为 256 个簇, 类似于你在作业 1 中所做的。这些将构成你的离散值观测序列 Y。
- 为单个字母定义隐马尔可夫模型 (HMM), 以及一个用于静音的 HMM。
- 通过串联单词拼写的 HMM 来构建每个单词的 HMM, 例如,$also \leftarrow [a][l][s][o]$ 和 $into \leftarrow [i][n][t][o]$。
- 使用训练样本来估计字母和静音 HMM 的参数。
- 使用得到的 HMM 集合来识别测试样本。
- 评估系统的识别准确性。

1.1 训练和测试数据文件

- **clsp.lblnames** 包含 256 个两个字符长的标签名, 每行一个, 共 256 行, 文件顶部有一个标题行。[总计:257 行]
- 有四个训练数据文件和两个测试数据文件, 如下所述。
- **clsp.trnscr** 是由语音组成训练数据的发言人朗读的脚本。词汇表中的 48 个单词以某种随机顺序呈现给发言人。脚本文件包含每个单词至少 10 个、最多 25 个样本, 总计 798 行数据, 文件顶部有一个标题行。[总计:799 行]

- **clsp.trnwav** 包含与上述脚本文件中每个单词的发音对应的语音 (波形) 文件的名称。这个文件也有 798 行, 顶部有一个标题行。[总计:799 行]
- **clsp.trnlbls** 包含 Y, 即与上述脚本文件中每个单词发音相对应的已处理语音。每行一个长标签字符串, 这个文件也有 798 行, 顶部有一个标题行。[总计:799 行,106,785 个标签]
- **clsp.endpts** 包含端点信息, 即关于每个说出的单词周围前导和尾随静音的信息。这些信息以每行两个整数的形式编码, 比如 i 和 j, 表示前导静音的最后一个标签在位置 i, 尾随静音的第一个标签在位置 j, 语音对应于标签文件中的第 (i+1) 到 (j-1) 个标签。同样有 798 行数据, 加上一个标题行。[总计:799 行]
- **clsp.devwav** 包含与测试集中每个单词的发音对应的语音 (波形) 文件的名称。这个文件有 393 行, 顶部有一个标题行。[总计:394 行]
- **clsp.devlbls** 包含标签 Y, 每行一个可变长度的标签字符串, 对应于测试集中每个单词的一个发音。这个文件有 393 行, 顶部有一个标题行。[总计:394 行,52,812 个标签]
- 只有 **clsp.trnscr**、**clsp.trnlbls** 和 **clsp.endpts** 用于 HMM 训练; 只有 **clsp.devlbls** 用于测试。**clsp.trnwav** 和 **clsp.devwav** 中列出的语音 (*.wav) 文件是为了让你听个乐趣。

1.2 构建主要识别器的步骤

1. 为每个字母创建 HMM: 这个词汇表中的单词不使用 [k]、[q] 和 [z]。对于剩下的 23 个字母, 创建一个 3 状态的 HMM, 具有左右拓扑, 对应于转移概率矩阵

$$\begin{bmatrix} 0.8 & 0.2 & 0.0 \\ 0.0 & 0.8 & 0.2 \\ 0.0 & 0.0 & 0.8 \end{bmatrix}$$

其中每个弧 (5 个) 上附加一个发射分布, 最后一个状态上剩余的 (0.2) 转移概率对应于一个用于退出此 HMM 的非发射 (空) 弧。

注意, 每个发射分布都在大小为 256 的输出字母表上。使用 **clsp.trnlbls** 中训练标签的"unigram" 频率初始化所有发射分布。(这是否确保没有标签具有零概率?)

2. 为静音和非语音声音创建 HMM: 创建一个名为 SIL 的 5 状态 HMM, 用于对单词周围的静音 (或噪音) 进行建模, 其拓扑对应于转移概率矩阵

$$\begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 & 0.00 \\ 0.00 & 0.25 & 0.25 & 0.25 & 0.25 \\ 0.00 & 0.25 & 0.25 & 0.25 & 0.25 \\ 0.00 & 0.25 & 0.25 & 0.25 & 0.25 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.75 \end{bmatrix}$$

   其中每个这些 (17) 弧上附加一个发射分布, 最后一个状态上剩余的 (0.25) 转移概率对应于一个用于退出此 HMM 的非发射 (空) 弧。使用 **clsp.trnlbls** 中前导和尾随静音的训练标签的 unigram 频率初始化所有发射分布, 如 **clsp.endpts** 所示。如果需要, 给每个 unigram 计数加 1, 以确保没有标签获得零概率。

3. 创建基于字母的发音: 使用每个单词的拼写作为其" 发音", 即作为子词单元的序列, 其串联生成单词的声学。

4. 形成单词 HMM: 通过根据步骤 3 中的发音串联步骤 1 中的字母 HMM 来构造每个 48 个单词的 HMM。将步骤 2 的 SIL HMM 附加到这个复合 HMM 的开头和结尾, 以创建单词 HMM。注意, 即使你在构建单词 HMM 时使用了两个 SIL HMM 的副本, 并且可能使用了同一字母 HMM 的多个副本, 这些都应该被认为具有绑定的参数。它们在所有 48 个单词的单词 HMM 中的使用中也是绑定的。换句话说, 在收集第 2 章中等式 (27) 和 (28) 的计数时, 应该只有步骤 1 和 2 中的转移概率和发射概率那么多累加器。

5. 训练单词 HMM: 使用训练数据文件来训练所有单词 HMM。特别是, 从这一阶段开始忽略端点信息, 并将静音模型与字母模型一起训练。(端点信息仅用于步骤 2 中初始化静音 HMM)。训练数据由每个 48 个单词的 10-25 个标记组成, 可以视为独立实现。因此, 与项目 1 不同, 你将在每个 798 个发音上分别运行前向-后向算法。但是, 一组公共的计数器 $c(t)$ 和 $c(y_t)$ 将累积后验概率 $P(\tau_i = t)$, 你将只在完成所有 798

次前向-后向传递后才进行第 2 章中等式 (36) 和 (37) 的参数更新。具体来说,

(a) 将字母 HMM 从字典序编号为 1 到 23, 让 SIL HMM 编号为 24。让第 i 个 HMM 中的第 j 个弧表示为 t_{ij}。注意,$i = 1, \ldots, 23$ 时 $j = 1, 2, \ldots, 6$, 而 $i = 24$ 时 $j = 1, 2, \ldots, 18$。对于每个弧 t_{ij}, 建立一个计数器 c_{ij}, 并将其初始化为零。对于每个非空弧 t_{ij}, 建立一个大小为 256 的计数器数组 $c_{ij}[y]$, 其中 y 遍历标签, 并将它们初始化为零。空弧不需要 $c_{ij}[y]$。

(b) 共同对文件 **clsp.trnscr** 和 **clsp.trnlbls** 中的行进行排序, 以便一个单词的所有 10-25 个发音彼此相邻。这样, 你可以构造一次单词 HMM 的网格, 并将其用于该单词的连续训练发音 10-25 次, 每个发音一次前向-后向传递。训练发音会增加该单词 HMM 中使用的所有弧 t_{ij} 的计数器 c_{ij} 和 $c_{ij}[]$。

(c) 在完成所有 798 次前向-后向传递后, 根据等式 (1) 更新 23 个字母 HMM 的参数:

$$p(t_{ij}) = \frac{c_{ij}}{\sum_{j':L(t_{ij})=L(t_{ij'})} c_{ij'}} \text{ 和} p(y \mid t_{ij}) = \frac{c_{ij}[y]}{c_{ij}}, j = 1, \ldots, 5, \tag{1}$$

对于 $i = 1, \ldots, 23$。类似地, 更新字母 HMM 所有空弧的转移概率 $p(t_{i6})$; 与它们没有关联的发射概率 $p(y \mid t_{i6})$。以类似的方式更新 SIL 模型的参数。

6. 观察收敛性: 绘制 798 个训练发音的总对数似然性作为参数更新 (1) 迭代次数的函数。检查似然性何时停止增加。(10 次迭代就足够了, 还是像你在项目 1 中可能做的那样需要 100 次迭代?)

7. 测试 HMM 系统: 对于测试数据文件中的 393 个发音中的每一个, 在每个 48 个单词模型下计算声学 (标签字符串) 的前向概率, 并选择似然性最高的单词。

1.3 主要系统报告中需要提交的内容

⇒ 从上面的步骤 6 中, 训练数据对数似然性作为迭代次数的函数的图。

- 在语音识别中, 通常报告每帧平均对数似然性, 即总对数似然性除以训练数据中的声学观测数。

⇒ 对于测试数据中的每个发音, 最可能的单词的身份和一个置信度。

- 要计算置信度值, 将最可能单词的前向概率除以所有 48 个单词的前向概率之和 (包括最可能的单词)。

⇒ 你的源代码, 以及关于每个模块需要哪些文件 (在为项目提供的 7 个数据文件中) 以及运行训练和测试模块的命令行 (用法) 的大量文档。

- 你的代码应该期望这 7 个文件在当前目录中, 并且应该在运行最近版本 GNU/Linux 的 x86_64 机器上运行。

预计会有一个包含所有上述内容的 tarball, 有明显的文件名和 README。

1.4 对比系统

构建对比系统以探索主系统的替代方案。下面提供了一些建议, 但你可以自由创新这部分项目。

(i) 不使用步骤 5 中的所有 798 个训练发音, 而是依靠步骤 6 中的似然收敛来决定何时停止 HMM 训练, 你可能希望将 HMM 训练的迭代次数基于识别准确性。将训练数据随机划分为 80% 的保留部分和 20% 的保留部分, 注意在两者之间平衡单词标记的分布: 即目标是将每个单词实例的 20% 放入保留集, 将 80% 保留在保留集中。根据步骤 1-5 构建系统, 但只使用保留的数据。在步骤 5 的每次迭代之后, 根据步骤 7 中所述在保留数据上测试生成的系统, 并测量准确性。当准确性停止增加时停止 HMM 训练, 并注意需要多少次迭代。让这个数字为 N^*。现在将保留和保留数据结合起来, 重复步骤 1-5 和 7, 在步骤 5 中使用 N^* 次 HMM 训练迭代, 而不管步骤 6 中的对数似然发生什么。

(ii) 不使用步骤 3 中基于字母的发音, 你可以使用手工制作的发音词典 (例如 http://www.speech.cs.cmu.edu/cgi-bin/cmudict), 并为英语的每个音素创建 HMM。使用步骤 3 中这些基于音素的发音重做步骤 1-7。将步骤 6 的对数似然图与基于字母的发音的相应图进行比较。

为你的对比系统提交类似的报告 (见上面标有 ⇒ 的项目)。

1.5 提交要求

提交的内容应包括:

- 主要系统和对比系统的详细报告
- 所有相关的源代码, 以及如何运行代码的清晰说明
- 生成的图表和结果
- 对结果的讨论和分析

所有文件应打包在一个 tarball 中, 并以明显的方式命名。请同时提供一个 README 文件, 概述 tarball 的内容以及如何复现你的结果。

构建对比系统以探索主系统的替代方案。下面提供了一些建议, 但你可以自由创新这部分项目。

(i) 不使用步骤 5 中的所有 798 个训练发音, 而是依靠步骤 6 中的似然收敛来决定何时停止 HMM 训练, 你可能希望将 HMM 训练的迭代次数基于识别准确性。

将训练数据随机划分为 80% 的保留部分和 20% 的保留部分, 注意在两者之间平衡单词标记的分布: 即目标是将每个单词实例的 20% 放入保留集, 将 80% 保留在保留集中。

根据步骤 1-5 构建系统, 但只使用保留的数据。在步骤 5 的每次迭代之后, 根据步骤 7 中所述在保留数据上测试生成的系统, 并测量准确性。当准确性停止增加时停止 HMM 训练, 并注意需要多少次迭代。让这个数字为 N*。现在将保留和保留数据结合起来, 重复步骤 1-5 和 7, 在步骤 5 中使用 N* 次 HMM 训练迭代, 而不管步骤 6 中的对数似然发生什么。

(ii) 不使用步骤 3 中基于字母的发音, 你可以使用手工制作的发音词典 (例如 http://www.speech.cs.cmu.edu/cgi-bin/cmudict), 并为英语的每个音素创建 HMM。使用步骤 3 中这些基于音素的发音重做步骤 1-7。将步骤 6 的对数似然图与基于字母的发音的相应图进行比较。

为你的对比系统提交类似的报告 (见上面标有 ⇒ 的项目)。

2 Project 2 Description

The goal of this project is to build an isolated-word recognizer for vocabulary of 48 words. • The speech signal has been processed to ex-

tract spectral features from small (25ms) windows at a xed frame-rate (100 frames/sec), and the resulting feature-vectors have been quantized into to 256 clusters, similar to what you did in Homework 1. These will constitute your discrete-valued observation sequences Y. • Define hidden Markov models (HMMs) for individual letters, and an HMM for silence. • Compose the HMM for each word by concatenating the HMMs of its spelling for example, $also \leftarrow [a][l][s][o]$ and $into \leftarrow [i][n][t][o]$. • Use the training samples to estimate the parameters of the letter and silence HMMs. • Recognize the test samples using the resulting set of HMMs. • Evaluate the systems recognition accuracy.

Training and Test Data Files: There is a file enumerating the possible values of the 256 cluster labels, i.e. the (quantized) outputs of the acoustic processor. **clsp.lblnames** contains 256 two-character-long label names, one per line, resulting in 256 lines, plus a title-line at the top of the file. [Total: 257 lines] There are four training-data les and two test data les, as described below. **clsp.trnscr** is the script that was read by the speakers whose speech comprises the training data. Each of the 48 words in the vocabulary were presented to speakers in some randomized order. The script-file contains at least 10 and up to 25 examples of each word, for a total of 798 lines of data, plus a title-line at the top of the file. [Total: 799 lines.] **clsp.trnwav** contains the name of the speech (waveform) file corresponding to the utterance of each word in the script file described above. There are 798 lines, plus a title-line at the top of this file as well. [Total: 799 lines.]

clsp.trnlbls contains Y, the processed speech corresponding to the utterance of each word in the script file described above. There is one long label-string per line, and there are 798 lines, plus a title-line at the top of this file as well. [Total: 799 lines, 106,785labels.] **clsp.endpts** contains endpoint information, i.e. informationabout the leading- and trailing-silence surrounding eachspokenword. This information is encoded in the form of two integers per line,say, i and j, to indicate that the last label of the leading silence is at position i, the first label of the trailing silence is at position j, and the speech corresponds to the (i+1)-th through (j 1)-th labels in the label-file. There are,again, 798 lines of data,plus a title-line. [Total: 799lines.]

clsp.devwav contains the name of the speech(waveform) file corresponding to the utterance of each word in the test set. There are 393 lines, plus a title-line at the top of this file as well. [Total: 394lines.] **clsp.devlbls** contains the labels Y, one variable-length label-string per line,corresponding to an utterance of each word in the test set.There are 393 lines,plus one title-line at the top of this file as well. [Total: 394 lines,52,812labels.] **Only-clsp.trnscr**,**clsp.trnlbls** and **clsp.endpts** are used for HMM training;and only **clsp.devlbls** for testing. Thespeech(*.wav) file listed in **clsp.trnwav** and **clsp.devwav** are provided for your listening pleasure.

Proceed to build your primary recognizer as follows. 1.Create the HMM for each letter: The words in this vocabulary do not use[k], [q]and[z].For the remaining 23 letters,create a 3-state HMM,with a left-to-right topology corresponding to the transition probability matrix 08 02 00 00 08 02 00 00 08 where an emission distribution is attached to each of these (5) arcs, and where the remaining (02) transition probability on the last state corresponds to a non-emitting (null) arc for exiting this HMM. Note that each emission distribution is on an output alphabet of size 256. Initialize all the emission distributions using the "unigram" frequency of the training labels in **clsp.trnlbls**. (Does this ensure that no label has zero probability?)

2.Create an HMM for silence and non-speech sounds: Create a 5-state HMM named SIL for modeling the silence(or noise) surrounding the words,with a topology corresponding to the transition probability matrix 025 025 025 025 000 000 025 025 025 025 000 025 025 025 025 000 025 025 025 025 025 000 000 000 000 075 where an emission distribution is attached to each of these (17) arcs, and where the remaining (025) transition probability on the last state corresponds to a non-emitting (null) arc for exiting this HMM. Initialize all the emission distributions using the unigram frequency of the training labels from the leading- and trailing-silences in **clsp.trnlbls**, as indicated by clsp.endpts. If needed, add 1 to each unigram count to make sure that no label gets zero probability.

3. Create grephemic baseforms: Use the spelling of each word as its "pronunciation", i.e. as the sequence of sub-word units whose concatenation generates the acoustics of the word. 4. Form word HMMs: Construct an

HMM for each of the 48 words by concatenating the letter HMMs of Step 1 based on its baseform in Step 3. Append the SIL HMM of Step 2 to both the beginning and the end of this composite HMM to create the word HMM. Note that even though you used two copies of the SIL HMM, and may have used more than one copy of the same letter HMM in composing the word HMM, these should all be considered to have tied parameters. They are also tied across their use in word HMMs of all 48 words. In other words, when collecting the counts per equations (27) and (28) in Chapter 2, there should be only as many accumulators as there are transition and emission probabilities in Steps 1 and 2 above. 5. Train the word HMMs: Use the training data les to train all the word HMMs. In particular, ignore the end-point information from this stage onwards, and train the silence model together with the letter models. (The end-point information was only used for initializing the silence HMM in Step 2.) The training data consists of 10-25 tokens of each of the 48 words, which may be treated as independent realizations. Therefore, unlike Project 1, you will run the forward-backward algorithm separately on each of the 798 utterances. But a common set of counters c(t) and c(yt) will accumulate the posterior probabilities P(ti = t), and you will carry out the parameter updates of equations (36) and (37) in Chapter 2 only after you have completed all 798 forward-backward passes. Speci cally, (a) Number the letter HMMs lexicographically from 1 to 23, and let the SIL HMM be numbered 24. Let the j-th arc in the i-th HMM be denoted t_{ij}. Note that $j = 1, 2, ..., 6$ for $i = 1, ..., 23$, and $j = 1, 2, ..., 18$ for i = 24. For each arc t_{ij}, establish a counter c_{ij}, and initialize it to zero. 6 for For each non-null arc t_{ij}, establish a 256-sized array of counters $c_{ij}[y]$, where y runs over the labels, and initialize them to zero. No $c_{ij}[y]$ are needed for null arcs.

(b) Jointly sort the lines in the les **clsp.trnscr** and **clsp.trnlbls**, so that all 10-25 utterances of a word follow each other. This way, you can construct the trellis for a word HMM once, and use it 10-25 times for successive training utterances of a word, one forward-backward pass per utterance. A training utterance increments the counters c_{ij} and $c_{ij}[]$ of all arcs t_{ij} used in the HMM of that word. (c) After all 798 forward-backward passes are

completed, update the parameters of the 23 letter HMMs as equation (1):

$$p(t_{ij}) = \frac{c_{ij}}{\sum_{j':L(t_{ij})=L(t_{ij})} c_{ij'}} and p(y \mid t_{ij}) = \frac{c_{ij}[y]}{c_{ij}}, j = 1, ..., 5, \qquad (2)$$

for i = 1,..., 23. Similarly update the transition probability $p(t_{i6})$ for all the null arcs of the letter HMMs; no emission probability $p(y \mid t_{i6})$ is associated with them. Update the parameters of the SIL model similarly.

6. Observe convergence: Plot the total log-likelihood of the 798 training utterances as a function of the number of iterations of the parameter update (1). Check when the likelihood stops increasing. (Do 10s of iterations su ce, or do you need 100s of iterations, as you probably did in Project 1?) 7. Test the HMM system: Foreach of the 393 utterances in the test data- file, compute the forward-probability of the acoustics (label string) under each of the 48 word models, and pick the word with the highest likelihood.

Submit the following in your report for the primary system for this project.

⇒ A plot of the training data log-likelihood as a function of the number of iterations from Step 6 above. • It is customary in speech recognition to report the average per-frame log-likelihood, which is the total log-likelihood divided by the number of acoustic observations in the training data.

⇒ For each utterance in the test data, the identity of the most likely word and a con dence. • To calculate the condence value, divide the forward-probability of the most likely word by the sum of the forward probabilities of all the 48 words (including the most likely word).

⇒ Your source code, along with substantial documentation about exactly what files (among the 7 data files provided for the project) are needed to run each module, and the command line (usage) for running the training and testing modules. • Your code should expect the 7 files to be in the current directory, and should run on a x86_64 machine running a recent version of GNU/Linux.

A tarball containing all the above, with obvious lenames and a README are expected. Build a contrastive system to explore alternatives to your primary system. Some suggestions are provided below, but you are free to innovate for this part of the project. (i) Instead of using all 798 training

utterances in Step 5 and relying on the likelihood convergence of Step 6 to decide when to stop HMM training, you may want to base the number of iterations of HMM training on recognition accuracy. Randomly divide the training data into an 80% kept portion and a 20% held-out por tion, taking care to balance the distribution of word-tokens between the two: i.e. aim to get 20% of the instances of each word into the held-out set, retaining 80kept set. Build the system according to Steps 1-5, but using only the kept data. After each iteration of Step 5, test the resulting system on the held-out data as described in Step 7, and measure accuracy. Stop HMM training when the accuracy stops increasing and note how many iterations were needed. Let this be N*. Now combine the kept and held-out data and repeat Steps 1-5 and 7, using N* iterations of HMM training in Step 5 regardless of what happens to the log-likelihood in Step 6. (ii) Instead of using graphemic baseforms in Step 3, you may use a hand-crafted pronun ciation dictionary (e.g. http://www.speech.cs.cmu.edu/cgi-bin/cmudict) , and create HMMs for each phoneme of English. Redo Steps 1-7 with these phonemic baseforms in Step 3. Compare the log-likelihood plot of Step 6 with the corresponding plot for graphemic baseforms. Submit an analogous report for your contrastive system (see items marked $\Rightarrow$ above).