实验室 2：HMM 和您的工作

EECSE6870：语音识别

截止日期：2016 年 2 月 26 日下午 6 点

产品目录

1、工程项目概述

隐马尔可夫模型是当今几乎所有语音识别系统背后的一项基本技术。它们简单而优雅，但却非常强大。事实上，它们经常被认为是智能设计的证据，因为它们从更简单的概率模型如多项式或泊松分布自发进化而来是不可想象的。

本任务的目标是让学生在基于 HMM/GMM 的语音识别系统中实现基本算法，包括训练和解码的算法。为了简单起见，我们将使用单个高斯弧来建模 HMM 弧的输出分布，而不是高斯弧的混合分布，而我们使用的 HMM^将不会包含“跳过”弧（即，所有弧都有输出分布）。对于这个实验室，我们将使用孤立的数字话语（如在实验室 1 中）和连续的数字字符串。

实验室由以下部分组成，全部需要：

- 第 1 部分：实现维特比算法一给定一个训练模型，写出给出话语中最有可能的单词序列的算法。
- 第 2 部分：实现高斯训练一，实现了重新估计高斯训练的均值和方差的算法。
- **第 3 部分：实现前后算法——编写训练 HMM^所需的前后算法。**
- **第 4 部分：从头开始训练各种模型，并在不同的数字测试集上进行评估。一使用早期部分中开发的代码来运行一组实验。**

实验室所需的所有文件都可以在敏锐度/用户/e/lab2/目录中找到。在开始实验室之前，请阅读文件实验室 2.txt；这包括您在做实验室时必须回答的所有问题。关于实验室的问题可以张贴在广场上（可以通过课程作品访问）。

请在这个实验室中自由地使用广场，从我们之前给出的课程的时间来看，如果不是平凡的话。

2 第 1 部分：实现维特比算法

在这部分中，您将实现简单的 HMM 解码器^的有趣部分，即计算给出最可能的单词序列的程序。我们已经对由隔离数字组成的数据预先训练了 HMM 的观察概率，这是您将被解码的模型。

2.1 工作背景资料：

正如在第 5 次讲座（2 月 17 日）的幻灯片中所讨论的那样，我们可以使用一个 HMM 来建模一个词汇表中的每个单词。对于实验室，我们使用了与幻灯片中相同的 HMM 拓扑和参数化。也就是说，对于每个单词，我们计算它的标准发音有多少音素，并使用三倍的许多状态加上一个最终状态。我们线性排列状态，每个（非最终）状态对自身和下一个状态都有一个弧。我们将输出分布放在弧上（如幻灯片中），而不是状态（如读数中），每个输出分布都使用 GMM 进行建模。对于每个状态，其每个传出圆弧上的 GMM 都被视为相同的 GMM。

例如，考虑两个词，其标准发音可以写成两个音素 TUW。它的 HMM 包含 2x3=6 个状态和一个最终状态。直观地说，附加到第一状态的传出弧上的 GMM 可以被看作是建模与音素 T 的第一三分之一相关联的特征向量，第二个声明的 GMM 建模 T 的第二三分之一，等等。HMM 的拓扑可以被认为是容纳代表 T 第一三分之一的一个或多个特征向量，接着是代表 T 第二三分之一的一个或多个特征向量，等等。

现在，我们可以使用单词 HMM^来执行孤立的单词识别，就像使用 DTW 执行识别一样。我们没有有一个针对每个单词的训练示例（或模板），而是有一个单词 HMM。我们不计算每个单词和测试示例之间的特殊距离，而是计算每个单词 HMM 分配测试示例的概率，并选择分配最高概率的单词。我们可以使用动态规划有效地计算 HMM 分配给特征向量序列的总概率。

然而，这种方法不能很好地扩展^，我们可以使用另一种方法做得更好。我们可以单独评分每个单词 HMM，而是可以构建一个大的 HMM，由每个单词 HMM 并行粘在一起，并跟踪大 HMM 的每个部分属于哪个单词。然后，我们可以在这个大 HMM 上使用维特比算法，并进行回溯以恢复最高得分路径。通过看到 HMM 最好的路径属于哪个单词，我们就知道最好的单词是什么。理论上，在这个大 HMM 上进行动态编程的时间与所有单词 HMM^上的相应计算相同，但这种方法可以更好地剪枝，可以极大地加速计算。

此外，这种方法可以很容易地扩展到孤立的单词识别之外。

例如，考虑想要识别连续数字字符串而不是单个数字的情况。我们可以通过使用单位数的大 HMM，并简单地从最终状态回到开始状态添加一个圆弧来实现这一点。HMM 现在可以接受由多个数字组成的数字字符串，我们可以使用维特比算法来找到最好的字序列，就像我们对孤立的数字一样。在这种情况下，最佳路径可能会多次循环通过机器，从而产生几个输出字。我们使用“一个大的 HMM”ⁿ在这个实验室中的框架。

2.2 该解码器

对于实验室的这一部分，我们为您提供了大量的解码器，您必须填写一个关键部分。下面是解码器的概述：

- 加载到大 HMM 图中以用于解码以及相关的 GMMˆ。
- 对于要解码的每个声学信号：

  —使用 Lab1 的前端生成特征向量。

  —使用特征向量（*）对大 HMM 执行 Viterbi。

  —通过回溯来恢复最佳的单词序列。

你必须实现这个部分；我们做其他的事情。

现在，这里概述了您需要处理的主要数据结构。

2.3 最大的 HMM 图

我们将大 HMM 存储在 Graph 类型的结构中（单击查看文档）。一定要检查一下迭代一个状态的传出弧的示例，因为你需要这个，如果有点奇怪。（造成这种奇怪现象的原因与该类未来可能的实现和效率有关。）没有简单的方法来找到一个状态的输入弧，但你不需要这个实验室。（这也适用于前后算法，即使不清楚如何做。）

弧存储在类弧中。注意，过渡日志问题（通常为弧表示 aj 可以通过弧访问。get_log_prob()和观察日志问题(通常表示 bij(t)或一些此类的)可以通过 arc.get_gmm()获得。具体地说，arc.get_gmm()返回一组 GMMˆ中的 GMM 的整数索引，它保存在 GmmSet 类型的对象中。

一个 GmmSet 对应于一个声学模型，它包含一组 GMMˆ，每个原始声学单位都有一个，从 0 以上编号。为了计算一帧上电弧的观测概率，我们将计算相关特征向量的相应 GMM 的概率。然而，我们已经预先计算了每一帧的每个 GMM 的对数问题，并将其卡在矩阵 gmm 问题中，所以您所要做的就是从这里查找观察概率。顺便说一句，圆弧也可以持有一个单词标签，即 arc.get_word()；此信息需要恢复与 HMM 路径相关的字序列。

此外：除了有一个开始状态（所有合法路径都经常开始的状态），HMM 还可以有最终状态，声明合法路径必须结束。在我们的实现中，一个图可以有多个最终状态，每个最终状态都可以有一个“最终概率”，在计算路径结束的概率时乘以。但是，您不必担心这一点，因为我们提供了处理最终状态的所有代码。

2.4 动态编程原理图

当按照所有三个主要的 HMM 任务（似然计算、维特比、正向-反向）的要求进行动态编程时，您需要填写一个值矩阵，即正向概率或回溯指针等。这个值矩阵有时被称为动态编程图表，在图表的每个位置计算的值元组有时称为图表的单元格。适当地，我们定义了一个名为 VitCell 的结构（单击文档），它可以将您需要的值存储在 Viterbi 解码的单元格中，并在一个变量命名图表中为您分配一个单元格矩阵，供您填写 Viterbi 计算。对于 Viterbi，应填写图表中每个单元格的日志概率和回溯弧 ID。

需要注意的是，我们不会直接存储概率或概率，而是存储日志概率(基 e)。这是因为如果我们直接存储概率，我们可能会导致数值下流。更多信息，请阅读福尔摩斯书中的第 9.12 节（第 153 页）！！！我们不是在开玩笑：如果你在开始之前不理解第 9.12 节中的概念，你将会陷入一个痛苦的世界！！（这些阅读资料可以在本网站上找到。）例如，我们将为帧 0 的开始状态初始化日志问题初始化为日志问题值 0，因为 ln1=0。如果我们想设置一个值对应于概率 0，我们将将其设置为日志-x，或者我们提供的非常接近的常数 g_zeroLogProb。提示：你可能会试图将日志概率转换为常规概率，以使你的头脑更清楚，但要抵制诱惑！我们使用对数概率的原因是，转换为规则概率可能会导致下流。

2.5 你应该做什么

要准备练习，请创建相关子目录并复制所需的文件：

```
mkdir-p~/e6870/实验室 2/
chmod0700~/e6870/

cd~/e6870/lab2/
cp-d，用户，事实，e6870，lab2/*。
```

请确保对 cp 使用“-d”标志（以便正确复制符号链接）。

本部分中的工作是填写文件 lab2_vit 中标记 BEGIN_LAB 和 END_LAB 之间的部分。C. 读取此文件以查看需要访问哪些输入和输出结构。若要编译此程序，请键入

```
制作 lab2_vit
```

它生成可执行的 lab2_vit。(这个链接包含在几个文件中，包括 util。C，其中包括 I/O 例程和 GmmSet 和图类；front_end.o，它保存从实验室 1 编译的前端；和主。C，它包含一个最小的主（)函数。）第 2.2 节概述了该程序的工作内容。

为了进行测试，我们创建了一个解码图，并训练了一些 GMM's 适用于孤立的数字识别。此运行中使用的大 HMM 或解码图包括一个（可选）沉默 HMM，接着是每个数字 HMM，然后是另一个（可选）沉默 HMM。若要对单个孤立的数字话语运行 lab2_vit，请运行该脚本

```
lab2_pla.sh
```

您可以检查此脚本，以查看已向 lab2_vit 提供了哪些参数。除了将解码字序列输出到标准输出之外，它还将动态编程图表的内容（第一个日志概率，然后是弧 ID）输出到 pla.chart.每一行对应不同

的帧，每列对应不同的状态。目标输出可以在 pla 中找到。图表。参考文献。您应该尝试或多或少精确地匹配目标输出，模算术误差。在实验室的后面部分，您将使用此解码器对使用以下部分中开发的代码训练的模型进行解码。

lab2.txt 中的说明将要求您运行脚本 lab2_plb。sh，它在十个单位数话语的测试集上运行解码器。

3 第 2 部分：实施高斯培训

在实验室的这一部分中，您将实现重新估计高斯分布所需的统计数据收集以及实际的重新估计算法。这将包括在 gmm_util 中填写 GmmStats 类的几种方法。C. 这个实验室中所有的高斯“mixture 矿将只包含一个高斯”。注意，我们在这里使用对角协方差高斯矩阵，所以我们不需要为每个高斯矩阵存储一个完整的协方差矩阵，而只需要沿对角线的协方差。

在典型的训练场景中，您有一个由音频话语和相应的参考成绩单组成的训练集。迭代训练集，在每次训练迭代中，首先将所有训练计数重置为 0。然后，对于每个话语，您运行转发反向算法；这给您每帧每个弧的后验计数。给定该弧上的 GMM/高斯值的恒等式、该弧的后验计数和该帧的特征向量，您有足够的信息来更新该 GMM/高斯值的训练统计信息。(这对应于 GmmStats：：add_gmm_count ()的方法。）在迭代结束时，您使用训练统计数据来更新所有 GMM/高斯均值和方差(如果我们使用 GMM，以及混合权重'具有多个组件）。（这对应于 GmmStats：：重新估计()的方法。）至于在哪里找到更新方程，您可以查看 Holmes 第 152 页的方程（9.50）和（9.51）或 HAH 第 396 页的方程（8.55）和（8.56）。提示：在重新估计方差时，请记住使用新的均值。提示：请记住，我们使用了对角线协方差高斯值，所以你只需要沿着协方差矩阵的对角线重新估计协方差。提示：请记住，差异不是 a！

由于您要在下一部分才实现前后算法，而不是在实验室的这部分使用前后算法，我们使用的是“维特比”版本的算法。也就是说，我们没有考虑 HMM 中所有可能的路径来计算每帧每个弧的后验计数，而是使用维特比解码来找到 HMM 中最可能的路径。然后，我们将该路径中每个弧的后验计数对于对应的帧为 1。事实上，这也是一个有效的训练更新，因为训练集的可能性是保证（弱）增加。该算法是广义电磁动算法的一个实例。在实践中，维特比 EM 训练通常用于训练的后期，而不是完全向后训练，因为它的计算成本较低，而且因为一旦模型变得非常清晰，这两种算法之间就没有很大的区别，因为 FB 后验通常会非常清晰。

与以前一样，您在本部分中的工作是填写适当文件中标记 BEGIN_LAB 和 END_LAB 之间的部分(gmm_util。C).读取文件，查看需要访问哪些输入和输出结构。若要编译此程序，请键入 makelab2_train。

这个程序所做的是首先加载一个声学模型/一组 GMM。然后，对于每个输入的音频话语，它加载一个使用维特比解码计算的话语的最佳 HMM 路径（或对齐）的表示。特别地，我们通过列出路径中每一帧相应的 GMM/高斯值来表示一个对齐。然后，对于具有相应的 GMM/高斯索引和特征向量（以及后验计数为 1.0）的每一帧，都调用了使用 add_gmm_count()的方法。在训练迭代结束时，调用对()的重新估计方法。最后，将重新估计的高斯参数写入到一个输出文件中。目前，不要担心这些 GMM 到底是如何实现的's 符合我们的整体声学模型；我们[5]在下一部分将会讨论这个问题。

为了进行测试，我们创建了一个由大约 20 个孤立的数字话语组成的小训练集。在相同的训练集中使用使用 FB 训练的模型创建相应的对齐。初始模型，我们[5]11 用法是所有高斯均值和方差分别设置为 0 和 1。若要对此数据运行一次培训迭代，请键入

Iab2_p2a.sh

您可以检查此脚本，以查看已向 lab2_train 提供了哪些参数。忽略输出日志问题值（仅为 0）。此脚本将输出 GMM 参数到文件 p2a.gmm。文件的开头包含了一堆你可以忽略的东西；实际的高斯均值和方差都在文件的末尾。例如，您可以执行以下操作

尾翼-n20p2a.gmm

以查看文件的最后 20 行。每一行对应于一个高斯值，并为每个维数列出了平均值和方差。可在 p2a.gmm.ref.中找到目标输出您应该尝试与目标输出精确匹配，模算术错误。本部分的高斯重估计代码将在实验室的下一部分中实现的前后算法中使用。

lab2.txt 中的说明将要求您运行脚本 Iab2_p2b。sh，在同一数据集上运行培训器。

4 第 3 部分：实现防御防御算法

4.1 本计划

好吧，现在我们已经准备好应对前后训练了。前后训练的目标是创建良好的解码模型（即，第 1 部分）。摘要详细介绍了我们的模型中需要训练的所有参数。在这个实验室中，我们的解码词汇表由 12 个单词（一到九、零、哦和沉默）组成，总共由 34 个音素组成。我们对每个音素使用 3 个 HMM 状态（不包括最终态），给我们总共 102 个状态。这些状态都有两个我们需要分配概率的输出弧，和一个我们需要估计的 GMM/高斯弧。为了简单起见，我们认为[5]我们将忽略此部分中的转换概率（通过将它们全部设置为 1）。在实践中，过渡概率对性能的影响最小。为了估计我们的 102 个高斯人的参数，我们[5]请使用在最后一部分中完成的高斯训练代码。

若要查看第 1 部分中使用的高斯参数，请查看文件 p018k7.22.20.gmm. 做尾-n102p018k7.22.20.gmm，查看所有 102 高斯人的均值和方差。从单词到高斯指数的映射是任意的；即，单词 8 被分配给高斯 0 到 5，沉默词被分配给高斯 99 到 101。

现在，我们可以使用前后算法来估计这些参数。例如，我们可以任意初始化所有参数（即，0 表示，1 个方差）。然后，我们可以迭代我们的训练数据，在每次迭代后重新估计我们的参数，这样我们的模型分配给训练数据的概率就可以保证在最后一次迭代中增加（或至少不减少）。(我们也可以使用维特比 EM 算法，除了有一个鸡和鸡蛋的问题：我们如何提出一个模型来生成训练所需的初始对齐？)

我们在每次迭代中所做的操作如下：

- 将所有计数初始化为 0。
- 循环通过训练数据中的每个话语：

  —使用前端来生成特征向量。

  -加载一个 HMM，通过连接到话语参考记录中每个单词的 HMM。

  -在此 HMM 和声学特征向量上向前-向后运行，以更新计数。
- 使用这些计数来重新估计我们的参数。

现在，前后算法可以分解为两部分：计算每帧 Z 处每个弧 a 的后验概率，通常表示类似的东西，并使用这些 Y(a，t)值来更新您需要更新的计数。您已经在最后一节完成了后者，所以我们在这里所需要做的就是计算如何计算 7(a，t)，每帧每个弧的后验计数/概率。

4.2 你应该做什么

在本部分中，您将实现前后算法，以计算每帧中每个弧的后验概率。特别是，这对应于在文件 lab2_fb 中填写部分。C.

对于这个实验室，您只需要计算弧的后验计数，而不是状态，因为过渡概率和观测都位于这个实验室的弧上。这个会议与第 4 讲的幻灯片一致，但与阅读内容不一致，所以要小心。更准确地说，您需要计算值 c（S 当 S ‘，t）（这是 Y（a，t）值的另一个名称）（请参见第 4 节中的“估计模型参数”部分）。

具体地说，您需要编写代码来填充动态编程图表中每个单元格的正向（日志）概率。FbCell 类中保留一个单元格（单击查看文档）。

然后，我们提供代码来计算话语的总(log)概率，以及初始化图表中最后一帧的每个状态的向后(log)概率的代码。然后，您需要填写执行后验算法的代码，并计算后验计数。一旦您有了一帧中圆弧的后验计数 c(S 专 S ‘，t)，您必须将这个计数记录在 GmmCount 类型的对象中，并将其附加到向量 gmmCountList 中；代码中给出了如何这样做的示例。

在前后函数完成后，gmm 统计。gmm 计数列表中的每个元素调用 add_gmm_count（)来更新高斯训练统计数据(除了低于 0.001 的后验计数不会为了效率而被转发）。遍历整个训练集后，将调用 gmmStats.reestimate()以更新 GMM 参数。

提示:您可能会想要使用函数 add_log_probs()。在指定的阅读材料中阅读福尔摩斯家族的第 9.12.2 节（第 154 页），看看这是在做什么。注意：您应该在每个 GmmCount 对象中放置一个规则概率，而不是一个日志概率！

若要编译，请键入 makelab2_fb。要在一个孤立的数字话语上运行这个训练器，请运行

```
Iab2_p3a.sh
```

该脚本将每帧的前向概率矩阵、后向概率矩阵和每帧中每个 GMM 的后验计数矩阵输出给 p3a_chart.dat。每个矩阵中的每一行都对应于一个帧；对于 F+B 概率，每列映射到图中的不同状态，对于后端，每列映射到不同的 GMM。目标输出可以在 p3a_chart.ref 上找到。你应该尝试精确地匹配目标后验，模算术误差。（某些正向和向后项的值可以与目标不同，只要后端匹配，仍有正确的实现。）要可视化给定 GMM 的后验概率，可以运行八度值并键入类似的类型

```
加载 p3a_chart.dat
绘图（utt2a_post（：，100））
```

查看具有索引 99 的 GMM 的后端（与沉默词的第一个状态相关联的高斯语）。(这要求您运行 XWindows，并且您运行带有标志的“ssh”。)(Matlab 不会为此工作，除非你先咀嚼数据文件。)

为了测试训练是否正常工作，您可以通过键入大约 20 个单一数字话语的训练集中运行训练师进行一次迭代

Iab2_p3b.sh

这将输出重新估计的 GMM's 到 p3b.g 毫米。可以在 p3b.gmm.ref.上找到目标输出

一旦您认为一切正常工作，就运行该脚本

Iab2_p3c.sh

该脚本通过执行前后算法的 20 次迭代来重新估计高斯参数，并在每次迭代时输出平均对数问题/帧（如正向算法中计算的）。训练集是 20 个单位数的话语。如果 FB 算法的实现是正确的，这个日志程序应该总是（弱）增加，并且看起来像收敛。此脚本将训练的高斯参数输出到文件 p3c.gmm。最后，通过运行脚本，使用该声学模型解码一些（孤立的数字）测试数据

Iab2_p3d.sh

这应该与 lab2_p1b 的输出相匹配。因为为该部分提供的模型是在相同的训练集上训练的。

5 第 4 部分：从零开始培训各种模型，并在各种数字测试集上进行评估

在本节中，我们在一些更大的数据集上运行我们的训练器/解码器，除了孤立的数据之外，我们还查看连续的数字数据（由每个话语的多个连接的数字组成）。首先，如果是一个好主意重新编译你的程序，比如：

清洁
OPTFLAGS=-O2 制造 lab2_fblab2_vit

首先，让我们看看我们的 HMM/GMM 系统与我们在实验室 1 中开发的 DTW 系统的比较。我们创建了一个由 56 个测试扬声者中每个数字的 11 个独立数字组成的测试集，并对 56 个训练扬声器中的每个数字使用一个模板运行 DTW（为每个测试演讲者使用不同的训练扬声器）。由此产生的错误率为 18.8%。

运行以下脚本：

Iab2_p4a.sh，我穿了球，p4a.出局

首先用 100 个孤立的数字话语训练模型(有 5 次的 FB 迭代)，然后解码与上面相同的测试集；然后，训练 300 话语和解码的模型；然后，训练 1000 话语和解码的模型。解码后，脚本运行 p018h1。计算器。以计算解码输出的单词错误率。查看单词错误率如何根据训练集的大小而变化。训练后的模型被保存在以前缀 p4a 开头的各种文件中。

接下来，运行以下脚本：

Iab2_p4b.sh 我穿了球，p4b.出局

这采用了 Iab2_p4a.sh 输出的 300 话语模型，并使用该模型解码连接的数字字符串数据（而不是孤立

的数字）。它还在 300 个连接的数字序列上训练一个模型，并解码相同的测试集。

最后，我们包括了一个实验来观察过渡概率的影响。运行以下脚本：

Iab2_p4c.sh，我的发球台，p4c.出局

我们使用在由 Iab2_p4a.sh 产生的 100 个孤立的数字话语上训练过的模型进行解码，除了我们也包括训练过的转换概率，通过将它们嵌入到用于解码的大 HMM 中。

6 要交些什么

您应该在当前目录中拥有一个 ASCII 文件 lab2.txt 的副本。填写此文件中的所有字段，并使用提供的脚本提交此文件。