Data Mining and Machine Learning

Speech R g HTK

Peter Jančovič

Objectives

- Building an ASR system using Hidden Markov Model Toolkit (HTK)
  - Feature Representation
  - Training
  - Recognitio
- Introduction to Perl

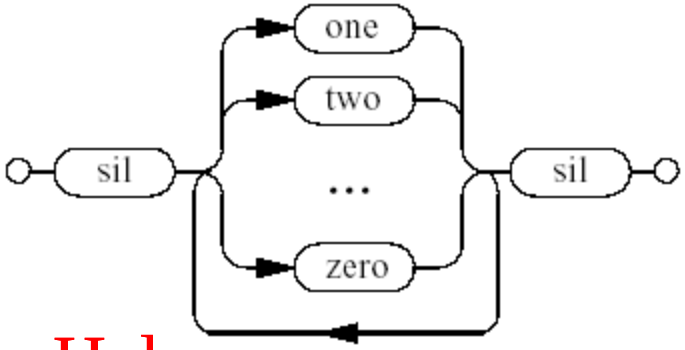
ASR system using HTK

- Hidden Markov Model toolkit (HTK) – available for free download at http://htk.eng.cam.ac.uk/
  - Set of tools – located in c:\HTK\HTK3.2bin
    - exe-files
    - manual fo
  - Tools likely to be used: HBuild, Rest, HInit, HList, HCopy, HRest, HLEd, HVite, H
  - Each tool called separately – passed input parameters, e.g., configuration files, list of files to be processed, etc.
- Chapter 3 in the HTK Manual (but phoneme-level)
- Connected digit ASR system

Task grammar

- Task grammar for voice dialing

Task grammar



- Task – connected digits recognition
  - Word-list file contains:

    one two … nine oh zero sil sp

- Create a text-fi (p.160 in HTK)


```
(
sil < one | two | three | four | five |
six | seven | eight | nine | zero > sil
)
```


- HParse.exe gram wdnet

```
VERSION=1.0
N=9   L=22
I=0   W=sil
I=1   W=one
I=2   W=two
I=3   W=three
I=4   W=sil
I=5   W=!NULL
I=6   W=!NULL
J=0    S=0    E=7
J=1    S=1    E=0
J=2    S=1    E=7
etc
```

Dictionary

- Dictionary for phoneme-level HMMs
  - Contains a list of words required in the task + their pronunciation, i.e., phoneme-level transcription
  - Example: / /v/
  - Create using HDMan tool
- Dictionary for word-level H
  - Pronunciation is the copy of the list of words
  - Example:

one	one
two	two

Data preparation

- Record data or use database provided
  - Training data – estimation of the parameters of the ASR system
  - Testing data – evaluation of the performance
- Label files – tr utterance – collected into Master La
  - Phoneme-level
  - Word-level
- Example: label_trainClean_noSP.mf contains:


```
#!MLF!#
"*/FAC_13A.lab“
Sil
One
Three
Sil
.
etc
```

- s

Feature extraction

- Extraction of speech acoustic features, e.g., MFCC, logFBE, LPC etc
- Use HCopy tool (Chs in HTK, p. 55-75)


```
# Coding parameters
TARGETKIND = MFCC_0
TARGETRATE = 100000.0
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 26
CEPLIFTER = 22
NUMCEPS = 12
ENORMALISE = F
```

List file
(.scp)

Creating word-level HMMs

- Training procedure
  - A set of single-Gaussian word-level HMMs
  - Start with a set of identical HMMs – means and variances are identica
  - Then perfor ns
  - Add short-pause (sp)
  - Loop: increase number of mixtures & perform several training iterations
  - Perform several final training iterations

Prototype HMM

- Define a prototype model – defines the model topology
  - number of states, covariance matrix type, feature type, feature dimension, nu
- Example: 8 state left-to-right HMM, no skips, diagonal covariance matrix, 1 stream, 39 dim feature vector

  Write a text-file containing:


```
<BeginHMM>
 <NumStates> 10 <VecSize> 39 <MFCC> <nullD> <diagC>
 <StreamInfo> 1 39
 <State> 2 <NumMixes> 1
  <Stream> 1
                1 1.0
                39
                0.0 …
                 9
                 0.0 …
                <NumMixes> 1

  <Mixture> 1 1.0
   <Mean> 39
    0.0 0.0 0.0 …
   <Variance> 39
    1.0 1.0 1.0 1.0 …
 <State> 4 <NumMixes> 1
  <Stream> 1
  <Mixture> 1 1.0
   <Mean> 39
…
```

Training – flat start (HCompV)

- Tool HCompV
  - compute the global mean and variance over the entire training data
  - set parameters of all of the Gaussians in a given HMM to these values
- HCompV.exe -m -S listTrain.scp 01 -M hmm0 prot
  - creates a new version of the prot ‘hmmdef’ in the directory ‘hmm0’
  - the zero means and unit variances replaced by the global speech means and variances
  - options: ‘-f’ – variance floor; ‘-o’ – output filename; ‘-S’ – file list

Training – creating initial HMMs

- Using ‘hmmdef”, construct HMM for all vocabulary units (digits, phonemes)
  - manually copying the “hmmdef” and relabeling it for each required di
  - automatically – write a sma n Perl or C (etc)
    - provided exe-files: macr els_1mixsil.exe

Training – HMM estimation (HERest)

- Tool HERest – estimation of the HMM parameters using Baum-Welch algorithm
- HERest -D -C $CONFIG -I $LABELS -t 250.0 150.0 1000.0 -S $LIST_FILE -H $HMM_DIR/hmm1/macros -H $HMM_DIR/hmm1/models -M $HMM_DIR/hmm2 $WORD_LIST

Training – HMM estimation (HERest)

- Perform several estimation iterations using the HERest
- Then generate 'short-pause' (sp) model
  - Copy the c odel
  - The 'sp' m dle state of the 'sil' model (HHEd tool used h
- Add the 'sp' in the last line of the WORD_LIST

Training – mixture increase (HHEd)

- Tool HHEd – various functions, including, increasing the number of mixtures
- Uses .hed file as input to define the function to be performed
- HHEd -H $HMM_DIR/hmm7/macros -H $HMM_DIR/hmm7/hmmdefs -H $HMM_DIR/hmm8/models -H $HMM_DIR/hmm9 $ED_CMDFILE2 $WORD_LISTSP
  - the file macros should contain the variance floor macro vFloors generated earlier

Recognition – HVite

- Tool HVite – performs recognition of an unknown utterance by using the Viterbi algorithm

Trained HMMs


```
#!MLF!#
"c:/Experiments/SpeechRecogHTK/dataAurora2/spec_ff3dct2a1/TESTA/CLEAN1/FAK_1B.rec“
0 2100000 sil -1527.106689
2100000 9100000 one -6118.945313
9100000 9200000 sp -74.889305
9200000 10900000 sil -1286.454468
.
"c:/Experiments/SpeechRecogHTK/dataAurora2/spec_ff3dct2a1/TESTA/CLEAN1/FAK_2B.rec“
etc
```

UNIVERSITY OF BIRMINGHAM

Recognition – HResults

- Compares the recout.mlf with the reference .mlf file – gives the recognition performance
- SENT: 197 of the 200 test utterances (98.50%) were correctly recognised
- WORD:
  - Indicates that o 9.77%) were recognised correctly
  - There was 1 deletion error (D), 1 sub) and 1 insertion error (I)
  - The accuracy figure (Acc) of 99.65% e percentage correct (Cor) because it takes account of the insert the latter ignores


```
====================== HTK Results Analysis ==============
Date: Sun Oct 22 16:14:45 1995
Ref : testrefs.mlf
Rec : recout.mlf
---------------------- Overall Results ----------------
SENT: %Correct=98.50 [H=197, S=3, N=200]
WORD: %Corr=99.77, Acc=99.65 [H=853, D=1, S=1, I=1, N=855]
============================================================
```

Introduction to Perl language

- Perl
  - programming language – text processing, e.g., files, strings
  - available o
- Creating and running a Pe
  - text file
  - Perl interpreter reads line by line and executes
  - run in the command prompt window

    > perl myprog.pl

Perl program

- Similar to C syntax
  - statements terminated by ;
  - comments
  - logical oper
- Variables
  - no need to pre-declare – variables are global


```
$x = 2;                    # variable 'x' will hold value 2
$greet = "hello";          # variable 'greet' will hold string
  'hello'
```

Perl program – Arrays

- Arrays

```
@array = (1, 2, “hello”);                    # a 3 element array
$x=1;
$y=2;
@nums = ($x+$y, $x-$y);                      # “nums” holds (3, -1)
$array[0] = $array[0] + $array[1];           # array[0] now holds 3
$len = @array;                               # variable ‘len’ holds 3 (the length of @array)
```


Perl program – Conditions

```
if (expr) {
    stmt;
}
else {
    stmt;
}
```

```
if (expr) {
    stmt;
}
```

```
if ($x > 3) { $x = 3; }
```


Perl program – Loops 1

```
while (expr) {
    stmt;
}
for (init_expr; t
    stmt;
}
for ($i=0; $i<100; $i++) {
    stmt;
}
```


Perl program – Loops 2

- Iterating over all elements of an array

```
foreach $var (@array) {
    stmt;
    @array
}
```


Perl program – External programs

- Running external programs
  - runs the HCopy.exe (from the HTK toolkit) with the given input parameters

system("HCop

Perl program – File operations, Print

- File handles to filenames as in C

```
open(F1, "filename");        # opens 'filename' for reading
open(F2, ">filename");       # opens 'filename' for writing
open(F3, ">>filename");      # opens 'filename' for appending
close(F1);
```


- Print output

```
print "Woo Hoo\n"           # prints a string to stdout
```

Perl program – Print output

- Example print output to a file

```
$fname = “file.txt”;
open(FILE, $fn      ) || die “Could not open $fname \n”;
print $FILE “S                                  intro.\n”;
```


- Perl Introduction based on

  http://cslibrary.stanford.edu/108/