Lecture 17

Continuous Speech Recognition

- Grammar Model
- Small Vocabulary Recogniser
- Pruning
- Continuous Speech Recognition
- Large Vocabulary Recognition
- Phoneme models
  - Triphones
  - Tied states

Grammar Model

- A grammar model defines the allowable word sequences
  - Each transition between words is given a probability
- We include a number of noise and silence models.
- Replace each word by the states that make up its model to give a single huge HMM
- If a word appears twice in the grammar, its model states must be duplicated.

Grammar for Telephone Dialler
2-word grammar example

Language Model:

HMM:

For each new frame, we use the Viterbi algorithm to find the best path that ends in each of the possible states.

The first word is definitely "left" since all paths converge when you trace them backwards in time.

The diagram is simplified: each word would actually have more states.

Small Vocabulary Recogniser

- One model for each word + silence, noise, cough etc
- Typically 10 to 15 states per word ⇒ <2000 states
- Need training examples of each word:
  - Single user (speaker dependent): >10 examples of each word
  - Many different users (speaker independent): >100 examples of each word
- Performance (with no grammatical constraints on word sequence)
  - Digits 0 to 9: <1% speaker independent error rate
  - Alphabet A to Z: <10% speaker independent error rate
    - This is a hard test because many letters sound similar: B,C,D,E,G,P,T & V
Pruning

At each frame find the log probability of the best path and remove any paths whose log probability is more than \( w \) less than the best.

This technique can remove a large fraction of the paths:
- Reduces recognition delay (e.g. all remaining paths start \( l \ r \ l \))
- Saves computation (fewer probabilities to check)
- The technique is known as a beam search: \( w \) is called the beam width

Continuous Speech Recognition

As each new frame arrives, we extend the lattice by one column.

Determine the possible predecessors for each state from the HMM diagram.

Select the predecessor with highest probability and calculate \( B(t,s) \) for each state.

Prune any paths whose probability is much lower than the best path.

Trace remaining paths backwards to find point of convergence.
Large Vocabulary Recognition

- Problems with recognising large vocabularies (>1000 words)
  - The searching task is now enormous: every alignment + word sequence ⇒ essential to have a good language model to limit the number of possible word sequences.
  - Very large memory requirements to store all the models.
  - More of the words are highly confusable ⇒ need to improve the acoustic modelling
    - Simple gaussian assumption is no longer good enough. Need $20F$ parameters to describe distribution instead of only $2F$.
  - Huge amount of training data required: 100s of hours of speech.
  - Must usually be able to cope with words that are not in the training data at all.

Phoneme Models

- All English words can be constructed from around 43 different phonemes:
  - Have a model for each phoneme
  - Construct a word or sentence by joining together the phonemes that make it up.
  - Some words may need alternative pronunciations:
    - “often” → a o f t n
    - “bath” → b a h
Uniphone Models

- *Uniphone* models have a single model for each phoneme
  - Typically 3 states per model
  - Total number of distinct states = $3 \times 43 = 129$
  - ⇒ very little training data needed
- Bad news:
  - Poor performance because the way in which a phoneme is pronounced depends on its context
    - Anticipatory coarticulation: speakers start to say one phoneme before they have finished the previous one ⇒ each phoneme is strongly affected by its neighbours.
    - In some cases phonemes can be affected by non-adjacent phonemes that occur later in the sentence but this is a smaller effect.

Triphone Models

- *Triphone* models have a $43^2$ different models for each phoneme: one for each possible combination of preceding and succeeding phoneme.
  - Some phoneme combinations never occur in English words so only about 60,000 phonemes are actually required
  - It is worth having whole-word models for a few common words whose pronunciation is non-standard: “the”, “and” etc.
  - 60,000 models × 3 states × 39 features × 20 parameters = 140,000,000 parameters to train.

"cat" → * → C → a → t → *

* denotes silence
Cross-word triphones

“cat sat”

* denotes silence
+ denotes inter-word gap

- Best results are obtained if triphones extend across words
  - We insert an optional “inter-word gap” between each word
  - Ignore the inter-word gap when considering which version of the triphone to use.

Training triphone models

- As with all HMMs, we start with a crude set of models and iteratively improve them:
  - Create a set of uniphone models: need some training data for which we know the beginning and end times of each individual phoneme.
    - For the rest of the training procedure, we only need to know what words are contained in each training sentence and not their start and end times.
  - Duplicate each of these monophone models 43^2 times to make an initial set of triphone models.
  - Repeat until convergence (Viterbi re-estimation):
    - Create a model for each of the training sentences by joining together the appropriate triphone models and then align each sentence with its model.
    - Re-estimate each state of each model by calculating the means and variances of all the feature vectors that were assigned to that state by the best alignment.
  - Repeat the previous step but this time use a weighted average of all possible alignments (Baum-Welch re-estimation).
Tied states

- Problem: there is always too little training data
  - Some triphones will occur very rarely, or even never, in the training data.
  - Solution: merge states that you expect to be similar
    - states that are tied together in this way combine their training examples and so get better estimates
  - can use phonetic knowledge to guide which states should be tied together
    - For example: the triphones shim and shin are followed by nasal consonants that are very similar so we can tie together their first two states.

Four versions of the phoneme "i":

pin
sin
shin
shim

State: i₁ i₂ i₃