The statistical approach to speech recognition

\[ W^* = \arg \max_W P(W \mid X, \Theta) \]

\[ = \arg \max_W \frac{P(X \mid W, \Theta) P(W \mid \Theta)}{P(X)} \]  
Bayes' rule

\[ = \arg \max_W P(X \mid W, \Theta) P(W \mid \Theta) \]  
P(X) irrelevant

- W is a sequence of words, W* is the best sequence.
- X is a sequence of acoustic features.
- \( \Theta \) is a set of model parameters.
Automatic speech recognition – Architecture

\[
W^* = \arg \max_W P(X | W, \Theta) \quad P(W | \Theta)
\]

Hidden Markov Model

- The state sequence is hidden. Unlike Markov Models, the state sequence cannot be uniquely deduced from the output sequence.

- In speech, the underlying states can be, say the identities of the sounds generating the features. These are hidden – they are not uniquely deduced from the output features. We already mentioned that speech has memory. A process which has memory and hidden states implies HMM.
Three problems of general interest for an HMM

3 problems need to be solved before we can use HMM’s:

1. Given an observed output sequence $X=x_1x_2..x_T$, compute $P_\theta(X)$ for a given model $\theta$

2. Given $X$, find the most likely state sequence (Viterbi algorithm)

3. Estimate the parameters of the model. (training)

These problems are easy to solve for a state-observable Markov model. More complicated for an HMM because we need to consider all possible state sequences.

How do we use HMMs to model words?

Simplest idea: whole word model

For each word in the vocabulary, decide on a topology (states and transitions)

Often the number of states in the model is chosen to be proportional to the number of phonemes in the word

Train the observation and transition parameters for a given word using examples of that word in the training data (Recall problem 3 associated with Hidden Markov Models)

Good domain for this approach: digits
Example topologies: Digits

- Vocabulary consists of {“zero”, “oh”, “one”, “two”, “three”, “four”, “five”, “six”, “seven”, “eight”, “nine”}
- The “longer” the word the more states needed in the HMM
- Must allow for different durations – use self loops and skip arcs
- Models look like:
  - “zero”
    - 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8
  - “oh”
    - 9 → 10

- “one”
  - 11 → 12 → 13 → 14 → 15 → 16

- “two”
  - 17 → 18 → 19 → 20

- “three”
  - 21 → 22 → 23 → 24 → 25 → 26

- “four”
  - 27 → 28 → 29 → 30 → 31 → 32

- “five”
  - 33 → 34 → 35 → 36 → 37 → 38
How to represent any sequence of digits?
Whole-word model limitations

- The whole-word model suffers from two main problems

  1. Cannot model unseen words. In fact, we need several samples of each word to train the models properly. Cannot share data among models – data sparseness problem.

  2. The number of parameters in the system is proportional to the vocabulary size.

- Thus, whole-word models are best on small vocabulary tasks.

Subword Units

- To reduce the number of parameters, we can compose word models from sub-word units.

- These units can be shared among words. Examples include:

<table>
<thead>
<tr>
<th>Units</th>
<th>Approximate number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phones</td>
<td>50</td>
</tr>
<tr>
<td>Diphones</td>
<td>2000</td>
</tr>
<tr>
<td>Triphones</td>
<td>10,000</td>
</tr>
<tr>
<td>Syllables</td>
<td>5,000</td>
</tr>
</tbody>
</table>

- Each unit is small

- The number of parameters is proportional to the number of units (not the number of words in the vocabulary as in whole-word models.)
Phonetic Models

- We represent each word as a sequence of phonemes. This representation is the “baseform” for the word.
  
  \[ \text{BANDS} \rightarrow B \text{ AE} N \text{ D Z} \]

- Some words need more than one baseform
  
  \[ \text{THE} \rightarrow DH \text{ UH} \]
  
  \[ DH \text{ IY} \]

---

Baseform Dictionary

- To determine the pronunciation of each word, we look it up in a dictionary
- Each word may have several possible pronunciations
- Every word in our training script and test vocabulary must be in the dictionary
- The dictionary is generally written by hand
- Prone to errors and inconsistencies

\begin{itemize}
  \item acapulco
  \begin{itemize}
    \item [AE K AX P AH L K OW]
    \item [AE K AX P UH K OW]
  \end{itemize}
  \item accelerator
  \begin{itemize}
    \item [AX K S EH L AX R EY DX ER]
    \item [IX K S EH L AX R EY DX ER]
  \end{itemize}
  \item acceleration
  \begin{itemize}
    \item [AX K S EH L AX R EY SH IX N]
    \item [AE K S EH L AX R EY SH IX N]
  \end{itemize}
  \item accent
  \begin{itemize}
    \item [AE K S EH N T]
    \item [AX K S EH P T]
  \end{itemize}
  \item accept
  \begin{itemize}
    \item [AX K S EH P T AX B AX L]
    \item [AE K S EH S]
  \end{itemize}
  \item acceptable
  \begin{itemize}
    \item [AX K S EH S AX R IY]
    \item [EH K S EH S AX R IY]
  \end{itemize}
  \item access
  \begin{itemize}
    \item [AX K S EH S AX R IY]
    \item [EH K S EH S AX R IY]
  \end{itemize}
\end{itemize}

2\textsuperscript{nd} looks wrong

AA K .. is missing

Shouldn’t the choices for the 1\textsuperscript{st} phoneme be the same for these 2 words?

Don’t these words All start the same?
Phonetic Models, cont’d

- We can allow for phonological variation by representing baseforms as graphs

  acapulco  AE K AX P AH L K OW
  acapulco  AA K AX P UH K OW

Now, construct a Markov model for each phone.

Examples:
Embedding

- Replace each phone by its Markov model to get a word model
- n.b. The model for each phone will have different parameter values

![Diagram of Markov model]

Reducing Parameters by Tying

- Consider the three-state model

![Diagram of three-state model]

- Note that
  - $t_1$ and $t_2$ correspond to the beginning of the phone
  - $t_3$ and $t_4$ correspond to the middle of the phone
  - $t_5$ and $t_6$ correspond to the end of the phone

- If we force the output distributions for each member of those pairs to be the same, then the training data requirements are reduced.
Tying

- A set of arcs in a Markov model are tied to one another if they are constrained to have identical output distributions.

- Similarly, states are tied if they have identical transition probabilities.

- Tying can be done in various ways

Implicit Tying

- Consider phone-based models for the digits vocabulary

0  Z  IY  R  OW
1  W  AA  N
2  T  UW
3  TH  R  IY
4  F  AO  R
5  F  AY  V
6  S  IH  K  S
7  S  EH  V AX  N
8  EY  T
9  N  AY  N

- Training samples of “0” will also affect models for “3” and “4”

- Useful in large vocabulary systems where the number of words is much greater than the number of phones.
Explicit Tying

Example:

6 non-null arcs, but only 3 different output distributions because of tying

Number of model parameters is reduced

Tying saves storage because only one copy of each distribution is saved

Fewer parameters mean less training data needed

Variations in realizations of phonemes

- The broad units, phonemes, have variants known as allophones

  Example: In English, p and pʰ (un-aspirated and aspirated p)
  Exercise: Put your hand in front of your mouth and pronounce “spin” and then “pin.” Note that the p in “pin” has a puff of air, while the p in “spin” does not.

- Articulators have inertia, thus the pronunciation of a phoneme is influenced by surrounding phonemes. This is known as co-articulation

  Example: Consider k and g in different contexts
  In “key” and “geese” the whole body of the tongue has to be pulled up to make the vowel
  Closure of the k moves forward compared to “caw” and “gauze”

  Phonemes have canonical articulator target positions that may or may not be reached in a particular utterance.
“keep”

“coop”
Allophones

- In our speech recognition system, we typically have about 50 allophones per phoneme.

- Older techniques ("tri-phones") tried to enumerate by hand the set of allophones for each phoneme.

- We currently use automatic methods to identify the allophones of a phoneme.

  - One way is a class of statistical models known as decision trees

Decision Trees

- We would like to find equivalence classes among our training samples

- The purpose of a decision tree is to map conditions (such as phonetic contexts) into equivalence classes

- The goal in constructing a decision tree is to create good equivalence classes
Decision Tree Construction

1. Find the best question for partitioning the data at a given node into 2 equivalence classes.

2. Repeat step 1 recursively on each child node.

3. Stop when there is insufficient data to continue or when the best question is not sufficiently helpful.

Decision Tree Construction – Fundamental Operation

- There is only 1 fundamental operation in tree construction:
  
  Find the best question for partitioning a subset of the data into two smaller subsets.

  i.e. Take an equivalence class and split it into 2 more-specific classes.
Decision Tree Greediness

- Tree construction proceeds from the top down – from root to leaf.
- Each split is intended to be locally optimal.
- Constructing a tree in this “greedy” fashion usually leads to a good tree, but probably not globally optimal.
- Finding the globally optimal tree is an NP-complete problem: it is not practical (and not clear it matters)

Pre-determined Questions

- The easiest way to construct a decision tree is to create in advance a list of possible questions for each variable.
- Finding the best question at any given node consists of subjecting all relevant variables to each of the questions, and picking the best combination of variable and question.
- In acoustic modeling, we typically ask about 10 variables: the 5 phones to the left of the current phone and the 5 phones to the right of the current phone. Since these variables all span the same alphabet (phone alphabet) only one list of questions.
- Each question on this list consists of a subset of the phonetic phone alphabet.
Sample Questions

\{P\}
\{T\}
\{K\}
\{B\}
\{D\}
\{G\}
\{P,T,K\}
\{B,D,G\}
\{P,T,K,B,D,G\}

Simple Binary Question

- A simple binary question consists of a single Boolean condition, and no Boolean operators.

- $X_1 \ c\ S_1?$ Is a simple question.

- $((X_1 \ c\ S_1) \ && \ (X_2 \ c\ S_2))?$ Is not a simple question.

- Topologically, a simple question looks like:

  \[ \text{x}\text{c}\text{S}? \]

  \[ \text{Y} \rightarrow \text{N} \]
Complex Binary Question

- A complex binary question has precisely 2 outcomes (yes, no) but has more than 1 Boolean condition and at least 1 Boolean operator.

- $(X_1 \ c \ S_1) \&\& (X_2 \ c \ S_2)$? Is a complex question.

- Topologically this question can be shown as:

```
  x_1cS_1?
     / \     \
    /   \   /   \
   /     \ /     \   \
  x_2cS_2? Y   N     Y   N
      |     |   |     |   |
      1     2   1     2
```

- All complex questions can be represented as binary trees with terminal nodes tied to produce 2 outcomes.

---

Configurations Currently Used

- All decision trees currently used in speech recognition use:

  a **pre-determined** set of simple, binary questions on discrete variables.
Tree Construction Overview

- Let $x_1 \ldots x_n$ denote $n$ discrete variables whose values may be asked about. Let $Q_{ij}$ denote the $j^{th}$ pre-determined question for $x_i$.

- Starting at the root, try splitting each node into 2 sub-nodes:
  1. For each variable $x_i$ evaluate questions $Q_{i1}$, $Q_{i2}$, … and let $Q’_i$ denote the best.
  2. Find the best pair $x_i$, $Q’_i$ and denote it $x’, Q’$.
  3. If $Q’$ is not sufficiently helpful, make the current node a leaf.
  4. Otherwise, split the current node into 2 new sub-nodes according to the answer of question $Q’$ on variable $x’$.

- Stop when all nodes are either too small to split further or have been marked as leaves.

Question Evaluation

- The best question at a node is the question which maximizes the likelihood of the training data at that node after applying the question.

- Goal: Find $Q$ such that $L(data | \mu_p, \Sigma_p) x L(data | \mu_r, \Sigma_r)$ is maximized.
Question Evaluation, cont’d

- Let $x_1, x_2, \ldots x_N$ be a sample of feature $x$, in which outcome $a_i$ occurs $c_i$ times.

- Let $Q$ be a question which partitions this sample into left and right sub-samples of size $n_l$ and $n_r$, respectively.

- Let $c_i^l, c_i^r$ denote the frequency of $a_i$ in the left and right sub-samples.

- The best question $Q$ for feature $x$ is the one which maximizes the conditional likelihood of the sample given $Q$, or, equivalently, maximizes the conditional log likelihood of the sample.

Best Question: Context-Dependent Prototypes

- For context-dependent prototypes, we use a decision tree for each arc in the Markov model.

- At each leaf of the tree is a continuous distribution which serves as a context-dependent model of the acoustic vectors associated with the arc.

- The distributions and the tree are created from acoustic feature vectors aligned with the arc.

- We grow the tree so as to maximize the likelihood of the training data (as always), but now the training data are real-valued vectors.

- We estimate the likelihood of the acoustic vectors during tree construction using a diagonal Gaussian model.

- When tree construction is complete, we replace the diagonal Gaussian models at the leaves by more accurate models to serve as the final prototypes. Typically we use mixtures of diagonal Gaussians.
Diagonal Gaussian Likelihood

- Let $Y = \{y_1, y_2, \ldots, y_n\}$ be a sample of independent $p$-dimensional acoustic vectors arising from a diagonal Gaussian distribution with mean $\mu$ and variances $\sigma^2$. Then

$$ \log L(Y \mid DG(\mu, \sigma^2)) = -\frac{1}{2} \sum_{i=1}^{n} \left\{ p \log(2\pi) + \sum_{j=1}^{p} \log \sigma_j^2 + \sum_{j=1}^{p} \frac{(y_{ij} - \mu_j)^2}{\sigma_j^2} \right\} $$

- The maximum likelihood estimates of $\mu$ and $\sigma^2$ are:

$$ \hat{\mu}_j = \frac{1}{n} \sum_{i=1}^{n} y_{ij} \quad j = 1, 2, \ldots, p $$

$$ \hat{\sigma}_j^2 = \frac{1}{n} \sum_{i=1}^{n} (y_{ij} - \hat{\mu}_j)^2 \quad j = 1, 2, \ldots, p $$

- Hence, an estimate of $\log L(Y)$ is

$$ \log L(Y \mid DG(\hat{\mu}, \hat{\sigma}^2)) = -\frac{1}{2} \sum_{i=1}^{n} \left\{ p \log(2\pi) + \sum_{j=1}^{p} \log \hat{\sigma}_j^2 + \sum_{j=1}^{p} \frac{(y_{ij} - \hat{\mu}_j)^2}{\hat{\sigma}_j^2} \right\} $$

Now,

$$ \sum_{i=1}^{n} (y_{ij} - \hat{\mu}_j)^2 = \sum_{i=1}^{n} y_{ij}^2 - 2\hat{\mu}_j \sum_{i=1}^{n} y_{ij} + n\hat{\mu}_j^2 \quad j = 1, 2, \ldots, p $$

$$ = \sum_{i=1}^{n} y_{ij}^2 - n\hat{\mu}_j^2 = n\hat{\sigma}_j^2 \quad j = 1, 2, \ldots, p $$

- Hence

$$ \log L(Y \mid DG(\hat{\mu}, \hat{\sigma}^2)) = -\frac{1}{2} \left\{ \sum_{i=1}^{n} p \log(2\pi) + \sum_{j=1}^{p} \sum_{i=1}^{n} \log \hat{\sigma}_j^2 + \sum_{j=1}^{p} n \right\} $$

$$ = -\frac{1}{2} \left\{ np \log(2\pi) + n \sum_{j=1}^{p} \log \hat{\sigma}_j^2 + np \right\} $$
Diagonal Gaussian Splits

- Let Q be a question which partitions Y into left and right sub-samples Y_l and Y_r of size n_l and n_r.
- The best question is the one which maximizes \( \log L(Y_l) + \log L(Y_r) \).
- Using a diagonal Gaussian model

\[
\log L(Y_l \mid DG(\hat{\mu}_l, \hat{\sigma}_l^2)) + \log L(Y_r \mid DG(\hat{\mu}_r, \hat{\sigma}_r^2))
\]

\[
= - \frac{1}{2} \{n_l p \log(2\pi) + n_l \sum_{j=1}^p \log \hat{\sigma}_l^2 + n_l p \}
\]

\[
+ n_r p \log(2\pi) + n_r \sum_{j=1}^p \log \hat{\sigma}_r^2 + n_r p \}
\]

\[
= - \frac{1}{2} \{np \log(2\pi) + np) - \frac{1}{2} \{n_l \sum_{j=1}^p \log \hat{\sigma}_l^2 + n_r \sum_{j=1}^p \log \hat{\sigma}_r^2 \}
\]

Common to all splits

Diagonal Gaussian Splits, cont’d

- Thus, the best question Q minimizes:

\[
D_Q = n_l \sum_{j=1}^p \log \hat{\sigma}_l^2 + n_r \sum_{j=1}^p \log \hat{\sigma}_r^2
\]

- Where

\[
\hat{\sigma}_j^2 = \frac{1}{n_l} \sum_{y_l \in y_l} y_j^2 - \frac{1}{n_l^2} (\sum_{y_l \in y_l} y_j)^2 \quad j = 1, 2, \ldots, p
\]

- \( D_Q \) involves little more than summing vector elements and their squares.
How Big a Tree?

- CART suggests cross-validation
  Measure performance on a held-out data set
  Choose the tree size that maximizes the likelihood of the held-out data
- In practice, simple heuristics seem to work well
- A decision tree is fully grown when no terminal node can be split
- Reasons for not splitting a node include:
  1. Insufficient data for accurate question evaluation
  2. Best question was not very helpful / did not improve the likelihood significantly
  3. Cannot cope with any more nodes due to CPU/memory limitations

Putting it all together

Given a word sequence, we can construct the corresponding Markov model by:

1. Re-writing word string as a sequence of phonemes
2. Concatenating phonetic models
3. Using the appropriate tree for each arc to determine which alloarc (leaf) is to be used in that context
Example

The rain in Spain falls ....

Look these words up in the dictionary to get:

DH AX | R EY N | IX N | S P EY N | F AA L Z | ...

Rewrite phones as states according to phonetic model

DH₁ DH₂ DH₃ AX₁ AX₂ AX₃ R₁ R₂ R₃ EY₁ EY₂ EY₃ ...

Using phonetic context, descend decision tree to find leaf sequences

DH₁ ₅ DH₂ ₂₇ DH₃ ₁₄ AX₁ ₅₃ AX₂ ₃₇ AX₃ ₁₁ R₁ ₄₂ R₂ ₄₆ ....

Use the Gaussian mixture model for the appropriate leaf as the observation probabilities for each state in the Hidden Markov Model.

Model Sizes

<table>
<thead>
<tr>
<th>Type</th>
<th>HMM</th>
<th>GMMs per State</th>
<th>GMMs</th>
<th>Gaussians</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
<td>Per word</td>
<td>1</td>
<td>10-500</td>
<td>100-10k</td>
</tr>
<tr>
<td>CI Phone</td>
<td>Per phone</td>
<td>1</td>
<td>~150</td>
<td>1k-3k</td>
</tr>
<tr>
<td>CD Phone</td>
<td>Per state</td>
<td>1-100</td>
<td>1k-10k</td>
<td>10k-600k</td>
</tr>
</tbody>
</table>
Recognition Performance

<table>
<thead>
<tr>
<th>System</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monophone</td>
<td>5.7</td>
<td>7.3</td>
<td>6.0</td>
<td>9.7</td>
</tr>
<tr>
<td>Triphone</td>
<td>3.7</td>
<td>4.6</td>
<td>4.2</td>
<td>7.0</td>
</tr>
<tr>
<td>State Based</td>
<td>3.1</td>
<td>3.8</td>
<td>3.4</td>
<td>6.3</td>
</tr>
</tbody>
</table>

- Resource Management (1K vocabulary task)