CHAPTER 2

Statistical Speech Recognition: A Tutorial

In this chapter, we provide a tutorial on statistical speech recognition. In particular, we establish hidden Markov models (HMMs) as a principal modeling tool for characterizing acoustic features in speech. The purpose of this chapter is to set up the context in which HMM parameter learning and discriminative learning in particular, will be introduced.

2.1 INTRODUCTION

A key to understanding the human speech process is the dynamic characterization of its sequential or variable-length pattern. Current state-of-the-art speech recognition systems are mainly based on HMMs for acoustic modeling. In general, it is assumed that the speech signal and its features are a realization of some semantic or linguist message encoded as a sequence of linguistic symbols. To recognize the underlying symbol sequence given a spoken utterance, the speech waveform is first converted into a sequence of feature vectors equally spaced in time. Each feature vector is assumed to represent the speech waveform over a short duration of 10–30 ms, wherein the speech waveform is regarded as a stationary signal. Typical parametric representations include linear prediction coefficients, perceptual linear prediction, and Mel frequency cepstral coefficients, plus their time derivatives. Furthermore, these vectors are usually considered independent observations given a state of HMM.

As illustrated in Figure 2.1, the role of speech recognizer is to map a sequence of observation vectors into its underlying words. Let the speech signal be represented by a sequence of observation vectors X,

$$X = x_1, x_2 \ldots, x_T$$

where x_t is the speech vector observed at time t. The speech recognition problem can therefore be regarded as looking for the most possible word sequence S^* given the observation vector sequence X, that is,

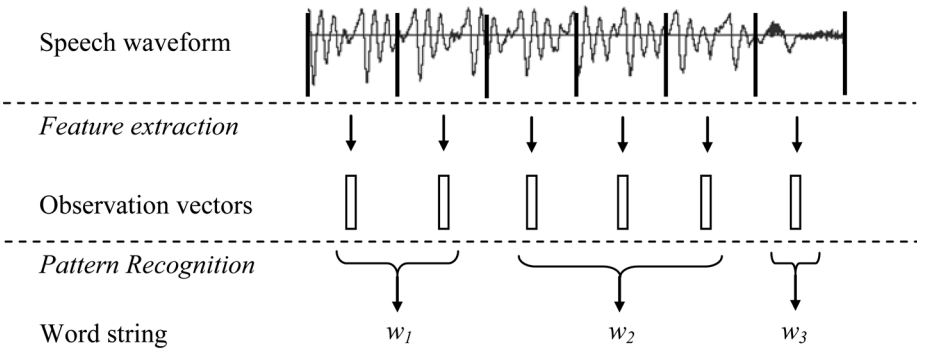

FIGURE 2.1: Illustration of the speech recognition process. The raw waveform of speech is first parameterized to discrete observation vectors. Then the word string that corresponds to that observation sequence is decoded by the recognizer.

$$S^* = \arg\max_s P(s|X) \tag{2.1}$$

According to Bayes rule, it is equivalent to solving S^* by:

$$S^* = \arg\max_s P(s,X)P(X) = \arg\max_s P(X|s)P(s) \tag{2.2}$$

where $P(s)$ is prior probability of the word sequence s, which is determined by a language model (LM), and $P(X|s)$ is the conditional probability or likelihood of X given s, which is computed from the acoustic model (AM) of the speech recognition system.

2.2 LANGUAGE MODELING

As described in the previous section, the a priori probability of the word sequence S is determined by the language model. For isolated-word speech recognition where recognition targets are isolated words. Given a K-word vocabulary, $P(w_i)$ is assigned to $1/K$ if a uniform distribution of word occurrence is assumed, or $P(w_i)$ can be determined by counting the occurrence frequency of word w_i in the language model training text corpus.

In continuous speech recognition, the computation of $P(S)$ is more complicated. Assume that the word sequence S has M words, that is,

$$S = w_1, w_2 \ldots, w_M$$

The probability of the word sequence S can be calculated as,

$$P(S) = P(w_1, w_2, \ldots, w_M) = P(w_1) \cdot \prod_{m=2}^{M} P(w_m | w_1, \ldots, w_{m-1}) \quad (2.3)$$

Assuming that the word sequence is produced by a $(N-1)$th order Markov model, the computation of $P(S)$ can be simplified as

$$P(S) = P(w_1)P(w_2|w_1)\ldots P(w_{N-1}|w_1, \ldots, w_{N-2}) \cdot \prod_{m=N}^{M} P(w_m | w_{m-N+1}, \ldots, w_{m-1}) \quad (2.4)$$

This model is referred to as an N-gram language model.

Many papers have been published on how to reliably estimate an N-gram language model. The basic idea is to count the frequency of occurrences of each word in the LM training text corpus, given a particular word sequence that precedes the word. To handle possible word sequences that are not seen in the training text, a back-off mechanism is normally used to assign lower-bound scores to those rarely seen word sequences. In most speech recognition systems, bigram and trigram LMs are used.

2.3 ACOUSTIC MODELING AND HMMs

In speech recognition, statistical properties of sound events are described by the acoustic model. Correspondingly, the likelihood score $p(X|s)$ in Eq. (2.2) is computed based on the acoustic model. In HMM-based speech recognition, it is assumed that the sequence of observed vectors corresponding to each word is generated by a Markov chain. For large-vocabulary automatic speech recognition (LVASR), usually an HMM is constructed for each phone, then the HMM of a word is constructed by concatenating corresponding phone-specific HMMs. We can further concatenate HMMs of words to construct the HMM of the whole string that contains multiple words. Then $p(X|s)$ is computed through $p(X|\lambda_s)$, where λ_s is the HMM of the strings.

As shown in Figure 2.2, an HMM is a finite-state machine that changes state once every time frame, and at each time frame t when a state j is entered, an observation vector x_t is generated from the emitting probability distribution $b_j(x_t)$. The transition property from state i to state j is specified by the transition probability a_{ij}. Moreover, two special nonemitting states are usually used in an HMM. They include an entry state, which is reached before the speech vector generation process begins, and an exit state, which is reached when the generative process terminates. Both states are reached only once. Because they do not generate any observation, none of them has an emitting probability density.

In the HMM, the transition probability a_{ij} is the probability of entering state j given the previous state i, that is, $a_{ij} = \Pr(q_t = j \mid q_{t-1} = i)$, where q_t is the state index at time t. For an N-state HMM, we have,

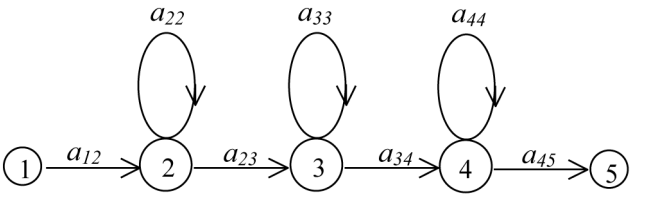

FIGURE 2.2: Illustration of a five-state left-to-right HMM. It has two nonemitting states and three emitting states. For each emitting state, the HMM is only allowed to remain at the same state or move to the next state.

$$\sum_{j=1}^{N} a_{ij} = 1 \tag{2.5}$$

The emitting probability density $b_j(x)$ describes the distribution of the observation vectors at state j. In discrete HMM (DHMM), emitting probability is represented by a multinomial distribution, whereas in continuous-density HMM (CDHMM), emitting probability density is often represented by a Gaussian mixture density:

$$b_j(x) = \sum_{m=1}^{M} c_{j,m} N\left(x;\mu_{jm},\Sigma_{jm}\right) \tag{2.6}$$

where $N(x;\mu_{jm},\Sigma_{jm}) = \dfrac{1}{(2\pi)^{\frac{D}{2}}|\Sigma_{jm}|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_{jm})^T\Sigma_{jm}^{-1}(x-\mu_{jm})}$ is a multivariate Gaussian density, D is the dimension of the feature vector x, and cjm, mjm, and Sjm are the weight, mean, and covariance of the mth Gaussian component of the mixture distribution at state j. Generally speaking, each emitting distribution characterizes a sound event, and the distribution must be specific enough to allow discrimination between different sounds as well as robust enough to account for the variability in natural speech.

Given $\{a_{ij}\}$ and $b_j(x)$, for $i = 1, 2, \ldots, N, j = 1, 2, \ldots, N$, the likelihood of an observation sequence X is calculated as:

$$p(X|\lambda) = \sum_{q} p\left(X, q|\lambda\right) \tag{2.7}$$

where $q = q_1, q_2, \ldots, q_T$ is the HMM state sequence that generates the observation vector sequence $X = x_1, x_2, \ldots, x_T$, and the joint probability of X and the state sequence q given l is a product of the transition probabilities and the emitting probabilities

$$p(X, q|\lambda) = \prod_{t=1}^{T} b_{s_T}(x_T) a_{s_T s_{t+1}} \tag{2.8}$$

where q_{T+1} is the nonemitting exit state.

In practice, (2.7) can be approximately calculated as joint probability of the observation vector sequence X with the most possible state sequence, that is,

$$p(X|\lambda) \approx \max_{q} p(X, q|\lambda) \tag{2.9}$$

Although it is impractical to evaluate the quantities of (2.7) and (2.9) directly due to the huge number of possible state sequences when T is large, efficient recursive algorithms such as forward–backward method and Viterbi decoding method exist for computing them [10, 43].