0.1 MAXIMUM LIKELIHOOD ESTIMATION EXPLAINED

Maximum likelihood estimation is a “best-fit” statistical method for the estimation of the values of the parameters of a system, based on a set of observations of a random variable that is related to the parameters being estimated. Note that the parameters being estimated are not themselves random variables. Rather, they are assumed to be unknown constants.

Given a set of observations $\boldsymbol{X}$ of a random variable X and possibly a set of observations $\boldsymbol{Y}$ of a conditioning random variable Y, the maximum likelihood estimate of a set of parameters ϕ is given by

$$\hat{\phi} = \mathrm{argmax}_{\phi}\{P(\boldsymbol{X}|\boldsymbol{Y};\phi, \lambda)\} \tag{1.1}$$

where $P(\boldsymbol{X}|\boldsymbol{Y};\phi, \lambda)$ represents the probability that samples of the random variable X take the value $\boldsymbol{X}$, given that the samples of the random variable Y have taken the value $\boldsymbol{Y}$. The terms ϕ and λ after the semicolon represent the parameters of $P(X|Y)$, the distribution of X conditioned on Y. The terms to the left of the semicolon are all random variables and the ones to the right are not random (and have no distributions associated with them).

Note that in Equation 1.1 $\boldsymbol{Y}$ and λ are optional. In the simplest estimation there may be no conditioning variables or additional parameters, and the maximum likelihood estimator would simply be given by

$$\phi = \mathrm{argmax}_{\phi}\{P(\boldsymbol{X};\phi)\} \tag{1.2}$$

In Equations 1.1 and 1.2 the probability of $\boldsymbol{X}$ with respect to ϕ is maximized. The solution obtained would be identical if any monotonically increasing function of $P(\boldsymbol{X}|\boldsymbol{Y};\phi, \lambda)$ were maximized instead of $P(\boldsymbol{X}|\boldsymbol{Y};\phi, \lambda)$. One particularly useful monotonic function is the logarithm. The logarithm has the simplifying property that it transforms the multiplication operation to an addition. Thus, if the elements of the set $\boldsymbol{X}$ were independent, we would get

$$\begin{aligned}\log(P(\boldsymbol{X}|\boldsymbol{Y};\phi, \lambda)) &= \log(P(X_1, X_2, \ldots, X_N|\boldsymbol{Y};\phi, \lambda)) \\ &= \log(P(X_1|\boldsymbol{Y};\phi, \lambda)P(X_2|\boldsymbol{Y};\phi, \lambda)\ldots P(X_N|\boldsymbol{Y};\phi, \lambda)) \\ &= \sum_{i=1}^{N}\log(P(X_i|\boldsymbol{Y};\phi, \lambda))\end{aligned} \tag{1.3}$$

where X_1, X_2, etc. are elements of the set $\boldsymbol{X}$. The maximum likelihood estimator now becomes

$$\hat{\phi} = \mathrm{argmax}_{\phi}\left\{\sum_{i=1}^{N}\log(P(X_i|\boldsymbol{Y};\phi, \lambda))\right\} \tag{1.4}$$

Maximization of Equation 1.4 is frequently simpler than maximization of Equation 1.1. The term $\log(P(\boldsymbol{X}|\boldsymbol{Y};\phi, \lambda))$ is referred to as the log-likelihood of $\boldsymbol{X}$, or simply as the likelihood of $\boldsymbol{X}$ and is represented as $L(\boldsymbol{X}|\boldsymbol{Y};\phi, \lambda)$.

The maximization given by Equation 1.4 results in an estimate for the parameter ϕ such that the resulting distribution $P(X_i|\boldsymbol{Y};\phi, \lambda)$ is the closest fit to the data set $\boldsymbol{X}$. We illustrate this by the following example.

Example: Consider a given set of seven samples from a random variable X:

$\boldsymbol{X} = \{X_1, X_2, \ldots, X_N\}$

The random variable is further assumed to have a Gaussian distribution with unit variance. The mean of this distribution μ is to be estimated from the sample $\boldsymbol{X}$. The log-likelihood of any sample X_i from a Gaussian distribution is given by

$$L(\boldsymbol{X};\mu) = \log\left(\prod_{i=1}^{N}\frac{1}{\sqrt{2\pi}}e^{-\frac{(X_i-\mu)^2}{2}}\right) = C - \sum_{i=1}^{N}\frac{(X_i-\mu)^2}{2} \tag{1.5}$$

where C is a constant resulting from the terms not involving μ. To maximize $L(\boldsymbol{X};\mu)$, we differentiate it with respect to μ and get

$$0 = \frac{d}{d\mu}L(\boldsymbol{X};\mu) = \sum_{i=1}^{N}(X_i-\mu) = \sum_{i=1}^{N}X_i - N\mu \tag{1.6}$$

leading to

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N}X_i \tag{1.7}$$

which leads us to the conclusion that the maximum likelihood estimate of the mean of a Gaussian random variable is merely the average of the observed samples.

The maximum likelihood estimation of the mean of a Gaussian random variable is illustrated pictorially by Figure 1.1.

In general the maximum likelihood estimate of any parameter of the distribution of any Random variable results in a distribution that fits the histogram of the observed samples most closely. Thus, the ML estimate is dependent only the observed samples. This can lead to errors in estimation since the observed samples may not be a fair representation of the distribution of the random variable. This usually occurs when the number of observed samples is very small.

1.2 HOW HMM-BASED AUTOMATIC SPEECH RECOGNITION (ASR) SYSTEMS FUNCTION

ASR systems are essentially pattern classifiers. In any such system all utterances of speech are modeled as sequences of sounds. These sounds may either be the phonemes in a language, words in that language, or larger units, depending on various factors. The complete set of sounds that the ASR system has to recognize constitutes the classes modeled by it. In this discussion, we assume without loss of generality, that the

Figure 1.1 Pictorial illustration of ML estimation of the mean of a Gaussian RV. The histogram represents the histogram of the observed samples. The curve in the first figure represents a Gaussian with mean μ_1. The curve in the second figure represents a Gaussian whose mean is the average of the observed samples. The third figure shows a Gaussian with mean μ_2. Clearly, the Gaussian in the figure to the center fits the histogram of the data most closely. The mean of this Gaussian is therefore the maximum likelihood estimate of the mean of the RV.

sound classes modeled by the system are words. The ASR system then classifies segments of speech as belonging to one of these words, thus identifying them.

In ASR systems, classification is not performed using the speech signal directly. Instead, the speech signal is tansformed into a sequence of *feature vectors*, or *parameter vectors*, and classification is performed using these feature vectors. The feature vectors most widely used used are cepstral coefficients, or variants of cepstra (such as PLP cepstra) derived from power spectra of short windowed segments, or *frames* of speech. Each feature vector thus corresponds to a frame of speech.

Let $\boldsymbol{S}$ represent the sequence of feature vectors derived from the utterance being recognized. ASR systems identify the sequence of words in that utterance using the optimal classifier equation

$$\hat{W} = \mathrm{argmax}_W \{P(\boldsymbol{S}|W)P(W)\} \tag{1.8}$$

where $\hat{W}$ is the recognized sequence of words in that utterance. $P(W)$ is the *a priori* probability that the word sequence W was uttered and is usually specified by a *language model*. For in-depth information on language models nd language modeling we refer the reader to [JELINEKS BOOK REF]. $P(\boldsymbol{S}|W)$ is the likelihood of $\boldsymbol{S}$ given that the W was the sequence of words uttered. It is termed as the acoustic likelihood of the data and is obtained from the probability distribution of all sequences feature vectors that could represent the sequence of words W. In HMM-based speech recognition systems this probability distribution of sequences is modeled by an HMM. The following section describes the hidden Markov model in greater detail.

HMM-based modeling of the distributions of sequences of vectors

In HMM-based recognition systems the mechanism that generates the sequence of feature vectors representing any word is modeled by an HMM. When generating the sequence, the generator is assumed to be in one of a finite set of states at any instant of time. Each state has a probability distribution function, referred to as the *state distribution* of that state, associated with it. The hidden Markov modeling paradigm assumes that to generate the feature vector at any instant, the generator draws a vector from the state distribution of the state it is in at that instant. The vectors that the generator draws from a state distribution are

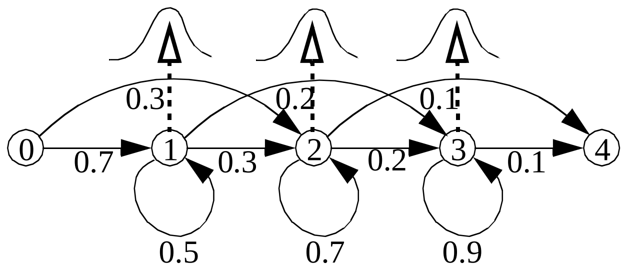

Figure 1.1 Example of a 5 state HMM with one non-emitting initial state, and a non-emitting terminating state. Each of the circles represents a state. The arrows represent valid transitions from the state, and the numbers below the arrows represent the probability of that transition. For example, the arrows from state 1 indicate that if the generator is in state 1 at time *t*, at time *t*+1 it can be in state 1 with probability 0.5, state 2 with probability 0.3 and state 3 with probability 0.2. The dotted arrows point to the state distributions associated with that state. An observation is drawn from this distribution every time the generator visits the state. The initial state (state 0) and the terminating state (state 4) have no state distributions associated with them, and no data are generated when the generator is in these states. Note that in this figure all transitions point left to right. In a more generic HMM, transitions may occur in any direction, from any state to any other state.

said to belong to that state. The HMM also has a set of transition probabilities associated with each state. A generator that is in state i at time t and moves to state j at time instant $t+1$ is said to *transit* from state i to state j at time instant t. The *transition probabilities* of a state refer to the probability distribution of the states that the generator can be in at the next instant, given that it is in that state at the current instant. The generator draws from this distribution in order to determine which state it will be in at the next instant of time. The transition probabilities and the state distributions are all specific to the word being modeled by the HMM. Figure 1.1 shows an example of an HMM with 5 states. The HMM in this figure only permits transitions in one direction. Transitions with probability 0 are not shown. This HMM has a non-emitting initial state and a non-emitting terminating state. Non-emitting states are states with which there are no probability distributions associated - the generator does not generate observations when it is in these states. The non-emitting initial state in figure 1.1 implies that at $t = 0$, *i.e.* just *before* the generator begins generating vectors, it is in the initial state where it does not generate any observations. Similarly, if the generator enters the terminating state it can no longer transit to any of the other states in the HMM, nor can it generate any more observations.

Thus, to generate a sequence of N vectors for the word, the generator is assumed to transit through a sequence of $N+2$ states in the HMM, beginning with the non-emitting initial state and terminating in the non-emitting final, or *absorbing* state. At each time instant it draws observations from the state distribution of the state it is in at that time instant. The sequence of vectors so generated is said to be *generated by the HMM*.

The model for the generating mechanism for a *sequence* of words is also an HMM and easily constructed by concatenating HMMs for individual words. Figure 1.2 shows an example where the HMMs for three words have been concatenated to obtain an HMM modeling a sequence of three words.

The statistical parameters of the HMM representing a sequence of words W are the set of transition probabilities, represented as a matrix A_W, and the set of state probability distribution functions. The matrix A_W consists of elements $a_w(i, j)$, each of which represents the probability that the generator will be in state j in the next time instant, given that it is currently in state i. Thus, for an HMM with K states, we have

$$\sum_{j=1}^{K} a_w(i,j) = 1.0 \tag{1.9}$$

The state distribution of the k^{th} state is represented by $P_{W,k}(X)$, where X represents any feature vector that belongs to that state. In speech recognition systems the various state distributions are usually modeled as Gaussians or mixtures of Gaussians. Typically, for computational efficiency, these Gaussians are assumed to have diagonal covariance matrices, *i.e.* covariance matrices where the off-diagonal elements are all 0. For simplicity we represent the state distribution of the k^{th} state as

$$P_{W,k}(X) = M(X;\phi_k^w) \tag{1.10}$$

where $M(X;\phi_k^w)$ denotes a Gaussian mixture distribution corresponding to the k^{th} state of the HMM representing the word sequence W and ϕ_k^w represents the set of parameters associated with it. We denote the set of ϕ_k^w for all the states in the HMM for W as λ_W. A_W and λ_W represent the complete set of parameters needed to uniquely identify the HMM modeling *W*.

The probability of any vector sequence $\boldsymbol{S}$ that is generated by the HMM for W is now given by

$$P(\boldsymbol{S}|W) = \sum_{\boldsymbol{s}\in\Xi} P(\boldsymbol{S},\boldsymbol{s}|W) = \sum_{\boldsymbol{s}\in\Xi} P(\boldsymbol{s}|W)P(\boldsymbol{S}|\boldsymbol{s}) \tag{1.11}$$

where $\boldsymbol{s}$ represents any state sequence that the generator can follow when generating $\boldsymbol{S}$, and Ξ represents the set of all possible state sequences. The state sequence $\boldsymbol{s}$ is, quite literally, a sequence of states, one for every feature vector in $\boldsymbol{S}$. That is,

$$\boldsymbol{s} = [s_1, s_2, s_3, \ldots, s_N] \tag{1.12}$$

where s_t is the state associated with $\boldsymbol{S}(t)$, the t^{th} vector in $\boldsymbol{S}$, and N is the total number of vectors in the sequence $\boldsymbol{S}$. The probability terms in the right hand side of Equation (1.11) can now be written as

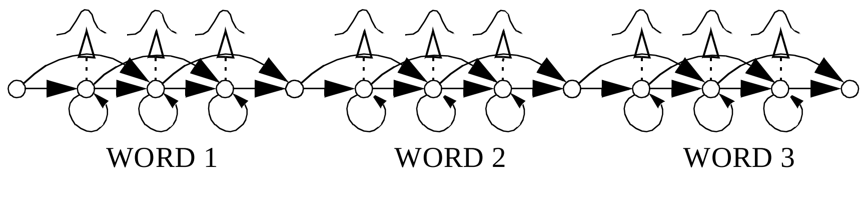

Figure 1.2 Example of constructing the HMM for a sequence of words from the HMMs of individual words. The non-emitting terminating state of any word is merged with the non-emitting initial state of the next word. The merged state is no longer an initial state or a terminating state. However, it remains non-emitting, and no state distribution is associated with it. The resulting HMM has a non-emitting initial state, a non-emitting terminating state and several intermediate non-emitting states as well.

$$P(\boldsymbol{S}|\boldsymbol{s}) = \prod_t M(\boldsymbol{S}(t);\phi_{s_t}^{w})$$

$$P(\boldsymbol{s}|W) = a(0,s_1)\prod_t a(s_t, s_{t+1}) \tag{1.13}$$

where $a(0, s_1)$ represents the probability of transiting from the 0^{th} state (i.e the initial non-emitting state) of the HMM for W to the first state in the state sequence $\boldsymbol{s}$, and $a(s_t, s_{t+1})$ represents the probability of transiting from state s_t to state s_{t+1}. Equation (1.11) can now be rewritten as

$$P(\boldsymbol{S}|W) = \sum_{\boldsymbol{s}\in\Xi, p}\left(a(0,s_1)\prod_t a(s_t, s_{t+1})\right)\left(\prod_t M(\boldsymbol{S}(t);\phi_{s_t}^{w})\right) \tag{1.14}$$

Ideally, recognition would be performed as

$$\hat{W} = \operatorname{argmax}_W\left\{P(W)\sum_{\boldsymbol{s}\in\Xi} P(\boldsymbol{s}|W)P(\boldsymbol{S}|\boldsymbol{s})\right\} \tag{1.15}$$

However, for easy implementation, HMM based speech recognition systems usually estimate not just the best word sequence, but also the best state sequence associated with the word sequence. *i.e.* recognition is performed as:

$$\hat{W} = \operatorname{argmax}_{W,\boldsymbol{s}}\{P(W)P(\boldsymbol{S},\boldsymbol{s}|W)\} = \operatorname{argmax}_{W,\boldsymbol{s}}\{P(W)P(\boldsymbol{s}|W)P(\boldsymbol{S}|\boldsymbol{s})\} \tag{1.16}$$

Using Equation (1.13), this can be further expanded into

$$\hat{W} = \operatorname{argmax}_{W,\boldsymbol{s}}\left\{P(W)\left(a(0,s_1)\prod_t a(s_t, s_{t+1})\right)\left(\prod_t M(\boldsymbol{S}(t);\phi_{s_t}^{w})\right)\right\} \tag{1.17}$$

In order to evaluate Equation (1.17) fully, the term within the braces would have to be computed for every possible word sequence in the language. This would be impractical. In practice, dynamic programming methods are used to obtain locally optimal estimates for $\hat{W}$.

There, is however, further refinement possiblein HMM-based systems. The CMU Sphinx-III HMM based recognition system, for example, is a phone-based recognition system in which words are further decomposed into sequences of phonetic units and the HMMs for words are built by concatenating the HMMs modeling these phoentic units. In order to reduce the total number of parameters needed to construct HMMs for all the phonetic units modeled by the system, the state distributions of the HMMs of the various phonetic units are shared, *i.e.* the same distribution is used by the states of the HMMs of several phonetic units.

In the next section we will explain the precise manner in which the Sphinx (or any other HMM-based system) models any given lanugage and learns the parameters of the models used to represent the basic sound units of the language.