**Hidden Markov models for speech recognition:**

The typical use of HMM's in speech recognition is not very different from the traditional pattern matching paradigm (Duda and Hart 1973). Successful applications of HMM methods usually involve the following steps:

1. Define a set of L sound classes for modeling, such as phonemes or words; call the sound classes V= {u1, u2. . . uL).
2. For each class, collect a sizable set (the training set) of labeled utterances that are known to be in the class.
3. Based on each training set, solve the estimation problem to obtain a "best" model λi for each class ***vi*** (i = 1, 2. . . *L).*
4. During recognition, evaluate Pr(**O** | λi) (I = 1, 2, . . . , L) for the unknown utterance **O** and identify the speech that produced **O** as class vi, if



The strengths of the HMM method lies in two broad areas: (1) Its mathematical framework and (2) its implementational structure.

The foundation of the HMM methodology is built on the well-established field of statistics and probability theory. That is to say, the development of the methodology follows a tractable mathematical structure that can be examined and studied analytically. The basic theoretical strength of the HMM is that it combines modeling of stationary stochastic processes (for the short-time spectra) and the temporal relationship among the processes (via a Markov chain) together in a well-defined probability space.

Another attractive feature of HMM's comes from the fact that it is relatively easy and straightforward to train a model from a given set of labeled training data (one or more sequences of observations). When the ML criterion is chosen as the estimation objective-that is. maximization of Pr(0 I i.) over i.-the well-known Baum-Welch algorithm is an iterative hill-climbing procedure that leads to, at least, a fixed-point solution as explained in Section 2.2. If we choose the state-optimized (or decoded) likelihood defined by



as the optimization criterion, the segmental k-means algorithm, which is an extended version of the Viterbi training segmentation algorithm, can be conveniently used to accomplish the parameter training task. The segmental k-means algorithm, as can be seen from the objective function of (17), involves two optimization steps-namely, the segmentation step and the optimization step. In the segmentation step, we find a state sequence q such that (17) is obtained for a given model *E.* and an observation sequence 0 . Then, given a state sequence tj and the observation 0, the optimization step finds a new set of model parameters 7so as to maximize (17): that is.



It can be rewritten as:



Note that  consists of two terms that can be separately optimized since is a function of only ***A,*** the state transition probability matrix, and is a function of only B, the family of (intrastate) observation distributions. This separate optimization is the main distinction between the Baum-Welch algorithm and the segmental k-means algorithm. These two training algorithms (Baum-Welch and segmental k-means) both result in well-formulated and well-behaved solutions. The segmental k-means algorithm, however, due to the separate optimization of the components of the model parameter set, leads to a more straightforward implementation. The ease of HMM training also extends to the choice of observation distributions. It is known that these algorithms can accommodate observation densities that are (a) strictly log-concave densities, (b) elliptically symmetric densities, (c) mixtures of distributions of the preceding two categories, and (d) discrete distributions. These choices of observation distribution in each state of the model allow accurate modeling of virtually unlimited types of data.

The flexibility of the basic HMM is manifested in three aspects of the model, namely: model topology, observation distributions, and decoding hierarchy. Many topological structures for HMM's have been studied for speech modeling. For modeling isolated utterances (i.e., whole words or phrases), we often use left-to-right models, since the utterance begins and ends at well-identified time instants (except in the case of very noisy or corrupted speech) and the sequential behavior of the speech is well represented by a sequential HMM. The choice of topological configuration and the number of states in the model is generally a reflection of the a priori knowledge of the particular speech source to be modeled and is not in any way related to the mathematical tractability or implementational considerations.



Two areas of concern in the implementation of any algorithm are the potential for numerical difficulties and the computational complexity. The potential numerical difficulties in implementing HMM systems come from the fact that the terms in the HMM probability measure of (7) and (8) are multiplicative. A direct outcome of the multiplicative chain is the need for excessive dynamic range in numerical values to prevent overflow or overflow problems in digital implementations. Numerical scaling and interpolation are two reasonable ways of avoiding such numerical problems. The scaling algorithm, alleviates the dynamic-range problem by normalizing the partial probabilities, such as the forward variable, at each time instance before they cause overflow or underflow. The scaling algorithm is naturally blended in the forward-backward procedure. Normalization alone, however, does not entirely solve the numeric problems that result from insufficient data support. Insufficient data support can cause spurious singularities in the model parameter space. One may resort to parameter smoothing and interpolation to alleviate such numerical singularity problems. A particularly interesting method to deal with sparse data problems is the scheme of deleted interpolation.

**HMM analogies to recognition of signatures and/or handwriting:**

**Averaged Bayes Classifier**

For a Bayes classifier ***e,*** its classification ***of*** an input ***x*** is actually based on a set of real value measurements-postprobabilities:



where *x Ci* denotes that *x* comes from class ***Ci.*** In the convention of statistical pattern recognition literature, these probabilities are simply denoted by *P(Ci/x), Vi*  A. They represent the probabilities that *x* comes from each of the *M* classes under the condition *x.* In principle, these probabilities are not related to each classifier ***ek .*** But in practice, that each ***ek*** classifies *x* is not really based on those true values of (1), which are not available. Instead, for each *x,* ***ek*** estimates by itself set approximations of those true values. These approximations depend on what features ***ek*** are used and how ***ek*** is trained. To clarify such dependence, we denote them as follows:



Now, we don’t care about the results of (3). Instead, we use the approximations of (2) for combining the classification results on the same *x* by all *K* classifiers. One simple approach here we propose is to use the following average value as a new estimation of combined classifier E:



post-probabilities. So, we call such a combined E as an averaged Bayes classifier. If we expect that the classified results are more reliable, we could use the following equation to replace (5) to take into account the trade-off between the substitution rate and the rejection rate in (6) (shown at the bottom of the page) with 0 1 being a threshold.

The previous approach could be extended to cover several cases when some ***ek’s*** belong to another kind of classifiers. First we consider the case that ***ek*** is a ***k*** - ***NN*** classifier. In this case, the classification process consists of two steps. The first one is to find the ***knn*** nearest prototype samples to the present input *x* with: 

where ***ki*** represents the number of prototype samples from class ***Ci.*** The second step is to classify *x* into class ***Cj*** according to ***kj*** = kithat is,



Since the measurements ki  i = 1, . . .M, have a different scale from the measurements in the form of post-probabilities, it is not reasonable to use (5) directly. The following formula introduces one way to transform ki ’s into the approximations as:



and then these approximations could be put into (5) for the subsequent combination computing. Second, we consider the case ***ek*** is some kind of distance classifier, i.e., for each **x, *ek*** classifies *x* according to some distance measures (e.g., Euclidean, Mahalanobis, and other pseudodistances etc.) ***dk(i)*** between *x* and the centers (or prototypes) of each class *Ci,* ***i*** = ***1, .... , M .*** If one could design some functions:



to derive a set of ***pk(i) ’ S*** which obey the three basic axioms of probability theory, one could use these ***pk(i)*** as apparent post-probabilities and put them into (4) for combination. Generally, any classifiers in which some kind of apparent post-probability are computable could be combined by means of (5).

**COMBINING MULTIPLE CLASSIFIERS BY VOTING PRINCIPLE**

The problem is to produce a new event *E(x)* = *j* from the given events ***ek(x)*** = ***jk , k*** = 1,. . , K, where the following equation may not necessarily hold: 

That is, conflicts may exist among the decisions of K classifiers. A simple and common rule used for resolving this kind of conflicts in human social life is voting by majority. This rule has been adapted for multi-classifier combination by [1], [2], [5] in the recognition of unconstrained numerals. In [5], eleven individual classifiers are proposed based on template matching, structural and statistical methods respectively. If six out of 11 vote for the same label, then the label is taken as the final result. In [2], two classifiers are used based on structural features extracted from skeleton and contour respectively. Three rules specific for the combination of the two classifiers are proposed. These rules may be regarded as the special examples of the majority voting rule. In [1], two classifiers are added to [2]. Among the four, each of the last three outputs only one label, i.e., e2(x) = j2 , e3(x) = j3, e4 (x) = j4; while the first one outputs a subset of labels, i.e., e1(x) = J with #|J| 3. The ek, k = 2, 3, 4 represents one vote that is assigned to its output label jk; while e1(x) divides one vote into #IJI fractions with each label in J receiving 1/(#lJl) vote. The decision is made such that the label that receives more than half of the votes (i.e., two) is taken as the final output.

Variants of Voting Principle and a General Expression

For convenience, we represent the event ek(z) = i in the form of a binary characteristic function:



The most conservative voting rule is the following



that is, the combined classifier E decides that x comes from Cj iff all the K classifiers decided that x comes from Cj simultaneously, otherwise it rejects x. In (15), “” denotes the operator of logical AND or binary multiplication, and in the following (15), “U” denotes the operator of logical OR or binary summation. A slight modification of (14) could lead to a version that is less conservative. The version is shown in (15) (at the bottom of the page), which results in an E that decides x Ci as long as some classifiers support x Ci and no other classifier supports a different x Cj, j i. Or in other words, (15) means that the classifiers that reject x have no impact on the combined E unless all the classifiers reject x.

The majority voting rule used in [1] could be expressed by the following formula:



where 0 < 1. Note that (16) is the special case of (18) with = 0.5 + , and > 0 is arbitrarily small. Equation (12) is equivalent to the special case of (18) with = 1.0. In (18), the thresholding operation only considers that the maximal votes of the final selected label must be large enough. There may exist cases that there are more than two labels that receive the maximal vote or the vote of the maximal is not considerably larger than the vote of the second maximal. In these cases, even the maximal vote of the final selected label may be quite large; the decision still may not be reliable since there exists an opponent that may also receive a large vote. To tackle this problem, a new majority voting rule is proposed in (19) (shown at the bottom of the page) and (20):



where 0 < 1. Since *K,* the number of classifiers, is constant, the votes of max2 could be regarded as the implicit objections to the label j. Thus, rule (19) in fact requires that the pure supports received by the finally selected label must be large enough. It is not difficult to see that rule (15) is equivalent to the special case of (18) with max2 = 0. All the aforementioned variants could be included in a general expression as



where dt(x) is a function of TE(x Ci), i = 1 ,...,M. Rule (17) is a special case of (21) with dt(x) = 0 and rule (20) is a special case of (21) with dt(x) = max2. dt(x) could also be another function, e.g., the median value of TE(z Ci), i = 1, ... , M. Observe that the threshold of (21) consists of two parts: a constant part that is independent of x, and a dynamic one that varies with input x. Finally, we should also point out that at the beginning of the subsection, although we only let Tk(x Ci) be the event of type ek(x) = i, i.e., each classifier outputs a single label. For the general case ek(x) = J with J being a subset of labels (e.g., expert no. 1 in [1]), all the previous equations also apply by defining a nonbinary characteristic function for the event ek(x) = J as follows:





