Lecture 5: Clustering and Adaptation

Lecturer: Mark Hasegawa-Johnson (jhasegaw@uiuc.edu)
TA: Sarah Borys (sborys@uiuc.edu)

January 12, 2009

Contents

1 Clustering: States with Similar Acoustic Spectra

The state index should encode all context information that might influence the acoustics: the third state of the /t/ in "a tree" should be different from the third state of the /t/ in "a tip," because they are acoustically different. Likewise, lexical stress, phrase position, glottalization, and dialect might matter.

Unfortunately, we never have training examples sufficient to learn the likelihood function associated with every possible combination of context variables. Therefore, we use a three-step strategy: (1) learn the mean and variance of MFCCs associated with every combination of context variables available in the training database, (2) cluster the context vectors, using acoustic similarity as a metric, ensuring that we have an adequate number of training examples from each phone, (3) learn a complete mixture Gaussian PDF for each clustered context-dependent phone.

There are two ways in which similarity-based clustering can be performed in HTK:

1. Bottom-up clustering is performed by the HHEd commands NC and TC. Bottom-up clustering searches through an arbitrary list of HMM states, finds the two states whose likelihood functions $b_i(\vec{o})$ and $b_j(\vec{o})$ are least distant from one another, and merges them to form a single PDF. The distance metric between mixture Gaussian likelihood functions is defined to be:

$$d(i,j) = -\sum_{k=1}^{K} \left[\log b_i(\mu_{jk}) + \log b_j(\mu_{ik})\right]$$

   Clusters are agglomerated using a farthest-distance heuristic, i.e., the distance between any two clusters S_1 and S_2 is defined to be

$$d(S_1, S_2) = \max_{i \in S_1, j \in S_2} d(i,j)$$

2. Top-down clustering is performed by the HHEd command TB, using splitting questions defined by the HHEd command QS. Top-down clustering results in trees like the one shown in Fig. 1. Top-down clustering starts with a group of states all merged together into a single root node. A long list of splitting questions (defined by you, the programmer, using the QS command) is then applied to the root node, and the one that best improves the log likelihood of the training database is selected to split the root node. The change in log likelihood is difficult to calculate in general, but very easy to calculate if the TB command requires all states to have a Gaussian (not mixture Gaussian) PDF; in that case, the change in log likelihood is

$$d(S_1, S_2) = N_1 \log|\Sigma_1| + N_2 \log|\Sigma_2| - (N_1 + N_2) \log|\Sigma_0|$$

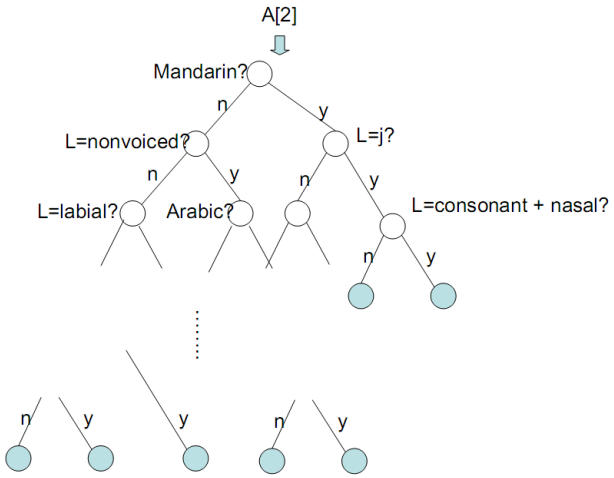

Figure 1: HHEd performs similarity-based acoustic clustering in order to group the allophone variants of each monophone, but does not allow different monophones to be grouped together. This constraint allows HTK to preserve a unique mapping from the dictionary to a triphone sequence. In the example shown here [1], each monophone comprises a large number of language-dependent triphone models; the clustering algorithm builds, for each monophone, a classification tree that determines, at each branch point, whether language or triphone context has the greatest impact on the acoustic spectrum. Language-ID questions ask, e.g., is the phone part of a Mandarin utterance? Triphone context questions ask, e.g., is the left-context phoneme labial?

where N_i is the number of frames in the database assigned to node i of the tree by the HERest command, $|\Sigma_1|$ and $|\Sigma_2|$ are the determinants of the covariance matrices of nodes 1 and 2, and the covariance of the parent node is related to the covariances of the child nodes by

$$\begin{aligned}(N_1+N_2-1)(\Sigma_0+\mu_0\mu_0^T) &= (N_1-1)(\Sigma_1+\mu_1\mu_1^T)+(N_2-1)(\Sigma_2+\mu_2\mu_2^T)\\ (N_1+N_2)\mu_0 &= N_1\mu_1+N_2\mu_2\end{aligned}$$

The TB algorithm then applies the same set of tests to each child node, and continues splitting in this way until the total occupancy of a node (N_i) is lower than some pre-determined number of frames, or the change in log-likelihood drops below some threshold.

It is usually a bad idea to combine, into a single cluster, the context-dependent allophones of different phonemes. For example, if one merged the triphones `/t->+p/` and `/t-@+p/` (worldbet notation), one would lose the ability to distinguish the words "top" and "tap." For this reason, the questions and trees defined by the QS and TB commands, respectively, and the clusters defined by NC and TC, are usually designed to create trees like the one shown in Fig. 1. Here the root of the tree contains all states with a certain position (first, second, or third) in an HMM whose center monophone has a particular label (/A/ in Fig. 1). The tree is then used to split up, into clusters, the context-dependent allophones of the specified monophone.

However, in some cases, one wants to merge HMMs that have different monophone labels. For example, you might want to figure out whether "cough" is most similar to "laugh" or to "fan_noise," because your training database does not have enough data to model each of these acoustic events distinctly. The HHEd syntax does not forbid this—each of the commands QS, TB, NC, and TC can contain an arbitrary list of HMMs—but previous versions of HTK have, nevertheless, been unable to grow a tree across multiple monophones. I haven't tested this using the current HTK release.

2 Clustering: States that Behave Similarly Under Adaptation

2.1 MLLR

Maximum likelihood linear regression (MLLR,[4]) adapts each of the Gaussians to a new talker by passing it through a linear transform:

$$\mu_{qk,adapted} = T_c \mu_{qk}$$

where μ_{qk} is the mean vector of the kth Gaussian in state q, and T_c is a linear transform matrix. The matrix T_c might be a global transform matrix (the same matrix is applied to all Gaussians), or it might be cluster-specific. If it is cluster-specific, then T_c is applied to μ_{qk} only if μ_{qk} is part of the cth cluster.

Regression class trees are computed in HHEd, using the command RC. The regression class tree is similar to the one shown in Fig. 1, with the following differences. First, the root node contains all Gaussians, not only the Gaussians associated with a particular monophone. Second, clustering can split groups of Gaussians, not groups of HMM states. Third, the distance between nodes is calculated differently.

A regression class tree should group together Gaussians whose mean vectors tend to behave similarly under speaker adaptation. That would be really easy to do, because regression coefficients computed from a Gaussian random vector are, themselves, Gaussian random vectors [2]. The mean and variance of each regression coefficient vector can, in principle, be easily computed from the training database, if one knows which training waveforms came from which talker.

Unfortunately, HTK currently does not compute regression class trees in this way (is anybody looking for an M.S. or Ph.D. thesis topic?)

Instead, the RC command in HHEd computes regression class trees based only on acoustic similarity, without explicitly using any information about the way the Gaussians behave under adaptation. The only important difference between the RC and TB command is that RC ignores covariances of the Gaussians it's clustering; instead, it performs K-means clustering of the mean vectors.

2.2 MAP

Speaker adaptation must somehow cope with the relatively small amount of training data available for a new speaker. Small-database training is performed using constrained or regularized learning. MLLR is a form of constrained learning (the adapted mean vector is equal to the input mean vector, transformed by a cluster-dependent linear transform). MAP (maximum *a posteriori* adaptation,[3]) is a form of regularized learning. For example, MAP learns the adapted mean vectors as a weighted sum of the speaker-dependent observation vectors $\vec{o}_t$, regularized by the addition of a regularization constant λ times the speaker-independent mean vector μ_{qk}:

$$\mu_{qk,adapted} = \frac{\lambda \mu_{qk} + \sum_t \gamma_{qk}(t) \vec{o}_t}{\lambda + \sum_t \gamma_{qk}(t)}$$

Unlike MLLR, MAP adaptation eventually converges (as the amount of training data for the new speaker grows) to a completely speaker-dependent HMM. However, for very small amounts of adaptation data, MAP tends to underperform MLLR. Many published works combine the techniques, using MLLR followed by MAP adaptation (both of which are implemented in HTK using HEAdapt), or other techniques (not provided in HTK) such as MAP-LR [5].

References

[1] Jui-Ting Huang, Xiaodan Zhuang, and Mark Hasegawa-Johnson. Language-independent phone models for multimedia search. speech recognition system developed for the Star Challenge competition, 2008.

[2] Richard A. Johnson and Dean W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice-Hall, Englewood Cliffs, NJ, third edition, 1992.

[3] Chin-Hui Lee, Chih-Heng Lin, and Bing-Hwang Juang. A study on speaker adaptation of the parameters of continuous density hidden markov models. *IEEE Trans. Speech and Audio Processing*, 39(4):806–814, 1991.

[4] C. J. Leggetter and P. C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Comput. Speech Lang*, 9:171–185, 1995.

[5] Olivier Siohan, Tor André Myrvoll, and Chin-Hui Lee. Structural maximum *a posteriori* linear regression for fast HMM adaptation. *Computer Speech and Language*, 16:5–24, 2002.