DIFFERENT STRATEGIES FOR DISTRIBUTION CLUSTERING USING DISCRETE, SEMICONTINUOUS AND CONTINUOUS HMMs IN CSR

Ricardo de Córdoba, José. M. Pardo

Speech Technology Group, Dpto. Ingeniería Electrónica, ETSIT Madrid, UPM
Ciudad Universitaria s/n. Madrid 28040. Spain
e-mail: cordoba@die.upm.es

ABSTRACT

We present an overview of different strategies and refinements to share parameters in HMM models at distribution (state) level for continuous speech recognition, showing the advantages and drawbacks of the different kinds of modeling [6]. We compare them with sharing at model level [5], achieving an error reduction close to 20% [4].

Discrete, semicontinuous and continuous HMM models are also compared using these approaches.

We consider two ways to smooth discrete distributions (interpolate detailed context dependent with robust context independent) derived from deleted interpolation [1] and coocurrence smoothing [10].

1. INTRODUCTION

In statistical modeling we need models that are as detailed and consistent (their repetitions are similar) as possible, but robust enough even if training data is limited. The choice of the units we will use is critical to achieve each of them. Typical consistent units are words, diphones and syllables, but usually they are poorly trained with large vocabularies. On the other hand, allophones are easily trained but they are not consistent, as they do not consider their context (context independent).

A solution for this problem is to use triphones, which are allophones that consider their right and left context, but again the problem is the large number of units we have. So, we need to share parameters between units. One classical solution is to use generalized triphones [7, 9] where similar triphone models are merged, as we did in [2, 5]. But it is more appropriate to share parameters at the state level [8], this way we do not merge models with different contexts, we just merge the states that are similar.

2. EXPERIMENTAL SETUP

We have used DARPA RM Speaker Dependent database, with 12 speakers, each one with 600 phrases for training and 100 for recognition. It is a medium vocabulary (about 1000 words).

We have used no grammar in our experiments, to better observe the improvements due to the acoustic modeling and not to the grammar. Of course, better results can be achieved using the word-pair. The result we are using in all our figures is the word accuracy, which includes the effect of the insertions.

The parametrization process is based on MFCC parameters. We use 10 MFCC parameters plus the total energy in one vector, and the differential and second differential parameters in a second and third vector.

3. DISCRETE MODELING

3.1. Base Model

Our base discrete model is a three state HMM model, with a total of 50 allophones, including three silence models. The model for the phrase is obtained concatenating the allophone models. We allow the transitions from one state to the second following one. To estimate the parameters we use the Viterbi algorithm. Results using this topology can be found in [2, 5] and Table 1.

3.2. Generalized Triphones

In this case, we share parameters at model level, merging the most similar triphones, creating a unit called generalized triphone. The basic algorithm is a typical clustering [9]: at first every triphone is a cluster, they are divided in classes according to their central allophone, and we merge the most similar pair of clusters until we have the desired number of clusters. For similarity we have used a distance based on smallest entropy increase, weighed by occurrence counts. We could have used other similar ones, but what it is important is to include both of them, to merge the less trained units first. For two distributions *a* and *b* the distance is

$$L(a,b) = N_{ab} H_{ab} - N_a H_a - N_b H_b$$

where N_a and N_b are the occurrence counts, $N_{ab} = N_a + N_b$, and H stands for entropy. For the complete model, we add up the contributions from all states and vectors of parameters.

We use context independent silence models to reduce the number of total models. We tested different number of final units and got an optimum for 350. You can see in Table 1 the global results averaging the twelve speakers. The improvement in error reduction is close to **15%** over context independent models.

3.3. Distribution clustering

The problem of the previous approach is that when we merge two units we do not consider if one of the contexts is similar, but the other one is not. It would be desirable that the units share just the context which is similar, and keep their own distributions when the contexts are different. This is the reason to cluster at state (or distribution) level.

The basic algorithm we used is similar to the one used in generalized triphones. This time we consider that each distribution is a cluster, and the distance measure is the same, based on smallest entropy increase and weighed by occurrence counts, but restricted to the distributions considered.

We include a restriction: we only merge distributions in the same position (left, right, or center) and belonging to the same central allophone, because it is quicker and we got better results (probably because many inter-state mergings were due to poor training). This way we can view the system as if we had a pool of distributions for each state and allophone. Any triphone unit chooses the most suitable distribution in the pool for each state.

To tune our algorithm we made the tests for one speaker. Our goal was to improve the result obtained using generalized triphones with that speaker (**69.7%** with 350 models), with a similar number of parameters. So, we began the clustering with about 7000 distributions and wanted to finish with about 1050 clusters.

We get an optimum using 1200 clusters, which is probably due to the fact that we can train more parameters using this approach with the same amount of training data. The value is 70.9% only but it will be increased with a few refinements.

The first modification in the algorithm is to give a bonus in the distance computing to clusters with similar contexts (we made classes of similar allophones for that, based in context independent models). We do not consider the training data to compute the value of the bonus. The purpose is to avoid wrong decisions caused by insufficient training. We implement that bonus as a weight between 0.6 and 1, averaging the similarity of the contexts of both clusters. We did not get a significant improvement, probably because the effect of this bonus is small when we only have 1200 clusters remaining.

Then, we introduced a change in our recognizer: use a different interword penalty for words and for interword silences (because we observed a lot of insertions of silences in the output). We found that the results showed stability as a function of this new variable we have introduced in our system, which is always desirable. The results improved to 76%, showing now that distribution clustering is much better than generalized triphones.

Another refinement we made was to allow the **swapping of distributions between clusters**, after each merge. The purpose of this approach is to avoid the progressive shift of a distribution from the "center" of a cluster as the clustering process progresses. It is possible that after a merge one of the distributions of a cluster gets closer to another cluster, so we would like to swap the distribution from one cluster to the other.

The problem of this approach is the large computational load involved: entropy measures are slow and after a swapping we have to compute all distances again. Our solution has been to introduce a preselection criterion based in average distance between distributions, which are precomputed values. So we only make additions and divisions, avoiding entropy computing. If we are considering to swap a distribution *i* from cluster *a* to *b*, we use:

$$\frac{d_b}{d_a} < Thresh_1$$

where d_a is the average distance from *i* to all the distributions in original cluster *a* and d_b is the average distance to the distributions in candidate cluster *b*. $Thresh_1$ is a threshold determined empirically, the bigger it is the less effective the preselection.

When this condition is satisfied, we consider if there is an improvement in entropy if the swapping is made. Here is the load of the algorithm, because we have to simulate the situation after the swapping and compute its entropy. If there is an improvement, we keep track of it, but the swapping will not be effective until all possible clusters are considered. Then we choose the swapping that gives the biggest improvement. Otherwise, distributions will make unnecessary swappings until they get to the best one. It is very important to keep track of all the computing done, because as the clustering process progresses many values are computed yet if the cluster considered has not changed.

To determine the right value for $Thresh_1$ we considered the relation between the number of times the preselection criterion is satisfied and the number of swappings actually made. We found in [4] that for a value of $Thresh_1 > 1.4$ the number of effective swappings began to saturate, so that is what we used. With all these restrictions, the global clustering process takes a CPU time of 3 or 4 hours in a Sparc 2 WorkStation, which is reasonable in a training stage.

In the recognition experiments, the results using this technique showed an improvement and were much more stable with the number of final clusters. The optimum was now for 1400 clusters, probably because with more clusters the swapping is much more effective, as there are many more clusters that are close but still keep differentiated. The results improved from 74.8% to **77.1%** (10% error reduction), using the same number of parameters.

In Table 1 we can see the global results for all speakers. We got a further improvement over generalized triphones close to **19%** in error reduction, using a similar number of parameters (a few more). Using the McNemar test all these figures have proven to be significant enough to state that the last system is the best.

Vectors of parameters	Context independent	Generalized triphones	Distribution clustering
2	59.3	66.6	73.1
3	61.1	66.0	72.1

Table 1: Global results for discrete systems

3.4. Smoothing the distributions

The purpose of smoothing is to balance the insufficient training data of context dependent discrete units. We have used two approaches:

1) **Approximation to deleted interpolation**

To interpolate a detailed context dependent distribution (CD) with its general context independent (CI) we use:

$$b_j = \lambda \cdot b_j^{cd} + (\quad) \cdot b_j^{ci}$$

where b_j is one element of the B matrix and λ is the weight for the context dependent unit, whose optimum we want to optimize.

The usual procedure is to divide training data in *n* blocks, do the training with *n-1* blocks and to optimize the probability in the other one (repeat for all the blocks). The problem is that it is expensive in CPU time. Our solution is to do the training as usual, and keep the accumulators of the models in each block in the last iteration. We decide the weights for the interpolation as a function of the accumulators -for context dependent and context independent- in one block and the result of merging the other *n-1* blocks [4]. It is a simplified algorithm but really fast. For all the speakers, results increased from 73.1% to **77.4%** (**16%** error reduction), which is a good figure for just a smoothing.

2) **Based in coocurrence** [10]

Using the coocurrence matrix of both context dependent and context independent distributions, we find the smoothed distribution. Then, this distribution is interpolated with the original one with a weight *(1-λ)* which is a function of the training data available for that unit (to use the CD when there is enough data to train it). The results are **77.4%** again, so using both approaches we get a similar improvement.

Finally, we included a uniform distribution in the interpolation, but results were slightly worse. It was probably because the threshold we use for the values of the B matrix is an equivalent effect.

4. SEMICONTINUOUS MODELING

4.1. Base Model

This type of modeling tries to overcome the drawbacks of:

- Discrete: lowering the quantization error and optimizing the mixture set along with the training
- Continuous: tying all pdf's we reduce the number of parameters and CPU time

We use a set of 256 gaussian mixtures common to all units (one for each vector of parameters), each with its mean vector and diagonal covariance matrix, and again with three states per unit.

We developed the Viterbi algorithm [5] for the training stage and introduced a preselection technique to optimize it in speed [3]. We found that it was better to use a fixed number of mixtures for each frame (4 or 5) not all of them [4, 5].

In Table 2 we can see the results using all speakers. There is an error reduction of **27%** over discrete models, which shows that this approach is really valuable.

4.2. Distribution clustering

For simplicity we use the same set of units obtained for discrete models. In Table 2 we can see the results for generalized triphones and distribution clustering. We must point out that there is an error reduction of **29%** in generalized triphones over context independent models, which is much bigger than the 15% we achieved with discrete modeling. The reason is that the effect of insufficient training data, always present in context dependent units, is smaller in semicontinuous models, because each frame contributes to more values (4 or 5 times in our case) reducing the number of non-trained parameters in the B matrix.

Another conclusion of this is that we do not need the smoothing step, because in this case the contribution of context independent models will not complete the context dependent unit.

Comparing distribution clustering with generalized triphones the improvement is close to **19%**, which is similar to the discrete modeling improvement. This is normal because they are both context dependent units and the effect of training data is similar in both of them.

Vectors of parameters	Context independent	Generalized triphones	Distribution clustering
2	69.6	78.2	82.3
3	72.2	80.5	84.1

Table 2: Global results for semicontinuous systems

5. CONTINUOUS MODELING

5.1. Multi-mixture context independent units

We have used the HTK tool [11] with the same MFCC parameters to compare with previous results. To speed up our experiments we used only one speaker of DARPA RM.

We model each state with a variable number of gaussian mixtures, each one with an mean vector and a diagonal covariance matrix (to reduce the number of parameters). We begin with one mixture per state and increase that number until fifteen using mixture splitting, to model the different possible contexts of each unit, reestimating the models in each stage. The results in Table 3 show that with ten mixtures it saturates (we have a behavior that is almost context dependent). It is obvious that if we want to improve the 80% result we have to guide the training using context dependent units.

5.2. Context dependent units

The best result so far with the speaker we are using in these experiments was 84.4%, which will be our goal. Our first approach was to use parameter tying [11], but our preliminary results were not encouraging enough (77% with one mixture per state), so we

Number of mixtures	Word accuracy	Iterations needed in training
1	53.1	5
5	73.7	10
10	79.0	11
15	80.2	12

Table 3: Results for context independent continuous system

decided to use distribution clustering as we did before. The clustering algorithm in HTK is similar to ours. It uses a distance measure based in the values of mean vectors and covariance, but the problem is that it does not consider the amount of training. To include that effect, we had to use a command, RO (minimum number of frames in training), which imposes the clustering of poorly trained distributions. Using RO=30, we get a result of **80.7%**, which is still under the semicontinuous value.

Then we increased the number of mixtures in each unit in different stages (to increase the number of parameters and model different variations of the units). What has proved to be interesting is to follow the next sequence [4]:

1. Increase in one the number of mixtures.
2. Continue with the clustering algorithm. The main purpose is to cluster a few units that are poorly trained when the number of mixtures is increased.
3. If there is no improvement or there are too many parameters, stop. Otherwise, go to 1.

The final number of mixtures for each unit was determined as a function of the amount of training data available for that state (between 3 and 7).

Our best result was **88%** using an average of 4 mixtures per unit and a total of 1217 units (giving about 5,000 mixtures in the global system). The result means an error reduction of **22%** over semicontinuous clusters. The reason is that the global number of mixtures is much larger, so there is more freedom for each unit to choose its appropriate mixture. The price we have to pay is that the processing time to estimate that number of mixtures increases.

6. CONCLUSIONS

- In all experiments state clustering was better than model clustering (about 20%).
- Best results with continuous modeling. The drawback is the CPU time, and the large number of mixtures in the global system.
- Semicontinuous is a good compromise between speed and results. We applied a preselection technique [3] to speed it up with good results.
- Discrete with smoothed models is the fastest one for a real-time system.

ACKNOWLEDGES

We have to thank Gabriel Ocaña for his assistance in the continuous modeling experiments and all the members of the Speech Technology Group for their advice.

REFERENCES

1. Bahl, L.R., F. Jelinek, R.L. Mercer. 1983. "A maximum likelyhood approach to continuous speech recognition". *IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 5, nº 2, pp. 179-190.*
2. Córdoba, R. de, J.M. Pardo, J. Colás. 1992. "Improving and optimizing speaker independent, 1000 words speech recognition in Spanish". *ICSLP Vol. I, pp. 161-164.*
3. Córdoba, R. de, X. Menéndez-Pidal, J. Macías-Guarasa, A. Gallardo, J.M. Pardo. 1995. "Development and improvement of a real-time ASR system for isolated digits in Spanish over the telephone line". *EUROSPEECH, vol. II, pp. 1537-1540*
4. Córdoba, R. de. 1995. *Sistemas de reconocimiento de habla continua y aislada: comparación y optimización de los sistemas de modelado y parametrización*. Ph.D. Thesis, ETSI Telecomunicación, Madrid.
5. Ferreiros, J., R. Cordoba, M.H. Savoji, J.M. Pardo. 1995. "Continuous speech HMM training system: Applications to speech recognition and phonetic label alignment". In NATO ASI Series "*Speech recognition and coding. New advances and trends*", pp. 68- 71. Ed. Springer Verlag.
6. Huang, X.D., H.W. Hon, M.Y. Hwang, K.F. Lee. 1993. "A comparative study of discrete, semicontinuous and continuous HMMs". *Computer Speech and Language, nº 7, pp. 359-368.*
7. Hwang M.Y., H.W. Hon, K.F. Lee. 1989. "Modeling Between-Word Coarticulation in Continuous Speech Recognition". *EUROSPEECH, pp. 5-8.*
8. Hwang, M.Y., X.D. Huang. 1991. "Acoustic Distribution Clustering in Phonetic HMMs". *EUROSPEECH, vol. 2, pp. 785-788.*
9. Lee, K.F. 1988. *Large-vocabulary speaker-independent continuous speech recognition: the SPHINX system*. Ph.D. Thesis, Carnegie Mellon University.
10. Schwartz, R., O. Kimbal, F. Kubala, M.W. Feng, Y.L. Chow, C. Barry, J. Makhoul. 1989. "Robust smoothing methods for discrete HMMs". *IEEE ICASSP, pp. 548-551*
11. Woodland, P.C., J.J. Odell, V. Valtchev, S.J. Young. 1994. "Large vocabulary CSR using HTK". *IEEE ICASSP, pp. II-125-128.*