A LOCALLY WEIGHTED DISTANCE MEASURE FOR EXAMPLE BASED SPEECH RECOGNITION

Mathias De Wachter, Kris Demuynck, Patrick Wambacq and Dirk Van Compernolle

Katholieke Universiteit Leuven – Dept. ESAT
Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium
{mathias.dewachter,kris.demuynck,patrick.wambacq,dirk.vancompernolle}@esat.kuleuven.ac.be

ABSTRACT

State-of-the-art speech recognition relies on a state-dependent distance measure. In HMM systems, the distance measure is trained into state-dependent covariance matrices using a maximum likelihood or discriminative criterion. This "automatic" adjustment of the distance measure is traditionally considered an inherent advantage of HMMs over DTW recognizers, as those typically rely on a uniform Euclidean distance. In this paper we show how to incorporate a non-uniform weighted distance measure into an example-based recognition system. By doing so we manage to combine the superior segmental behaviour of DTW with the near-optimal acoustic distance measure as found in HMMs. The non-uniform distance measure enforces modifications to the k nearest neighbours search, an essential component in our large vocabulary DTW approach. We show that the complexity of our solution remains within bounds. The validity of the full approach is verified by experimental results on the Resource Management and TIDigits tasks.

1. INTRODUCTION

Research into optimal acoustic features for ASR has shown that features are not equally informative for the different phones. Hence the strong acoustic dissimilarity between different sounds requires the acoustic feature vector to be an optimal compromise, rather than the optimal one for each sound independently.

In almost all state-of-the-art HMM speech recognition systems, an inherent automatic adjustment of the distance measure takes places at the level of the covariance matrices of each state. Indeed, this property is often considered an inherent advantage of HMMs over DTW, since traditionally DTW recognizers use a uniform Euclidean distance metric over a globally scaled acoustic space (e.g. through cepstral liftering).

In section 2, we show that this judgment is inaccurate, as the use of different covariance matrices for each HMM state can be translated to the example-based setting, by means of a *locally weighted distance measure*.

In section 3, the problems the non-uniformly scaled distance measure poses to the k-nearest neighbours selector of our prototype system are outlined. We propose a fast algorithm that can deal with non-uniform distance measures, and that takes advantage of the between-frame dependencies typically found in speech recognition tasks. Scaling experiments show its good behaviour with increasing database sizes.

Research funded by the Fund for Scientific Research Flanders (FWO-project G.0249.03).

2. A LOCALLY WEIGHTED DISTANCE MEASURE

2.1. Theoretical framework

Viterbi decoding for HMMs and DTW for example-based recognition are very similar. Both use a dynamic programming search to find a path through a finite state machine (FSM) that optimally matches a sequence of input vectors. And in both cases, the cost of the path consists of observation costs and transition costs. The former is used to match the current input vector to the FSM state and the latter to move from state to state. However, there are a few important differences, that can be summarized as follows:

- HMMs use a single model with complex local statistical distributions, while DTW uses a large amount of parallel templates with a simple distance metric.
- HMM emission probability densities are state-dependent, while DTW uses a uniform distance measure.

To understand similarities and differences better, it is useful to develop an HMM interpretation of our large vocabulary DTW recognizer by explicitly writing down the DTW recognizer as an HMM system. For the sake of simplicity, we will assume usage of unique sub-word units (e.g. phonemes) as elementary segments.

Each elementary unit is represented by all the examples in the database, and *each of those examples may be seen as an M-state HMM* where:

- every feature vector in the training database defines a unique HMM state.
- local constraints on state transitions –typical to DTW– are used to enforce temporal consistency of the input sequence and the template [1].
- observation costs are given by a single gaussian pdf. Its mean is the local feature vector and the covariance matrix is *somehow* defined by the distance measure.

Now, calling the D-dimensional input vector $\vec{x}$, the relevant database vector $\vec{y}$, and the applicable covariance matrix Σ, the negative logarithm of the observation likelihood is given by

$$\begin{aligned} -\log(f(\vec{x}|\vec{y})) &= \log\left((2\pi)^{D/2}\right) + \log\left(|\Sigma|^{1/2}\right) \\ &+ 1/2(\vec{x}-\vec{y})^T\Sigma^{-1}(\vec{x}-\vec{y}), \end{aligned} \quad (1)$$

which yields, dropping the constant first term and some constant scaling factors, an equivalent distance measure:

$$d(\vec{x};\vec{y}) = (\vec{x}-\vec{y})^T\Sigma^{-1}(\vec{x}-\vec{y}) + \log(|\Sigma|) \quad (2)$$

Eq. 2 consists of 2 terms: (a) a Mahalanobis distance between $\vec{x}$ and $\vec{y}$ and (b) a bias compensation. The covariance matrix parameterizes both terms.

A typical DTW system now uses a highly degenerative form of eq. 2 in which the covariance matrix is fixed to a single diagonal matrix for all states. This implies that the bias compensation term becomes identical for all instances and hence can be omitted. Furthermore, since the covariance weighting is uniform, it can be applied as a preprocessing step, and the resulting distance measure becomes the square of the L_2 metric. However, it should be stressed that this extreme simplification is *not necessary*, and below we will empirically show that it is far from optimal. Rather, it is supported by computational motives and ignorance about the covariance. On the other hand it leads to a true metric which is symmetric in $\vec{x}$ and $\vec{y}$, a property which is lacking in the HMM framework.

2.2. Associating covariances to the database vectors

The HMM framework requires the estimation of mean vectors and covariance matrices. In our example, though, each state has been defined by a single observation. Using the vector itself as mean seems reasonable, but there is no straightforward methodology to determine the covariance matrix. Grouping examples of the same phonemes into some "mean model" in order to have enough observations for covariance estimation obviously leads us back to a normal HMM approach. However, in this case the advantages of our example-based approach, which are a consequence of having all original information about each example available, would be lost [2]. Therefore, we keep all examples as separate models, and *indirectly* estimate suitable covariance matrices.

Let's state the facts and principles that will guide our further development:

- Within the HMM framework mean and variance are defined by state identity. Hence it is not the incidental value of $\vec{y}$ that should define Σ, but its underlying identity.
- From a pattern matching and metric perspective it would be nice if eq. 2 were symmetric in $\vec{x}$ and $\vec{y}$ and that Σ was a smooth function of all the parameters that define it.

The above criteria are not compatible and we will stick to the HMM approach for the time being. In first instance we approximate Σ in Eq. 2 with a diagonal covariance structure. Furthermore we assume that the underlying identity of reference vectors $\vec{y}$ can be clustered into M classes and we use the class' sample variance as an estimate for Σ.

Using these assumptions, we get a reduced form of eq. 2:

$$d(\vec{x};\vec{y}) = \sum_{j=1}^{D} \left(\frac{x_j - y_j}{\hat{\sigma}_{c,j}} \right)^2 + \log \left(\prod_{j=1}^{D} \hat{\sigma}_{c,j}^2 \right) \qquad (3)$$

with $\hat{\sigma}_{c,j}^2$ the sample variance of class c for dimension j, and c the class assigned to database vector $\vec{y}$. Of course, there still remains the problem of assigning class identities to each database frame.

As our "DTW-models" only have the basic acoustic modeling capabilities of a single shared diagonal-covariance Gaussian per class, we can accommodate for classes that are more fine-grained than HMM tied states. Clustering based on acoustic resemblance of vectors or complete templates, or based on discriminative criteria comes to mind.

Preprocessing	global	local weighting	Rel. Impr.
TIDigits			
cepstra (26 feat.)	1.46	0.90	38%
mida (15 feat.)	0.76	0.37	51%
Resource Management (feb89 test set, word-pair grammar)			
cepstra (26 feat.)	16.52	10.15	39%
mida (25 feat.)	9.92	6.21	37%

Table 1. *Word error rates and relative improvement for our prototype example-based recognizer. Two different types of preprocessing are combined with global Euclidean distance and the locally weighted distance.*

2.3. Experiments

In our proof-of-concept implementation, we only use class labeling information that is readily available from our in-house HMM system. The labels consist of both phonetic properties and non-verbal information. The phonetic properties are summarized as context dependent HMM state numbers based on a phonetic decision tree [3] for the RM task, and state numbers of context independent word-model HMMs for the TIDigits task. The non-verbal information consists of gender, dialect region and speaker identity. The basic recognition units are context independent phoneme examples for the RM task, and (also context independent) complete digit examples for the TIDigits task. Note that classes for assigning distance measures and recognition units are independent.

Classes are based on the product of HMM state number and non-verbal information. Since not enough data is available for each of these fine-grained classes, the variances are estimated using a hierarchical Bayesian re-estimation procedure, which handles smoothing and back-off. For the given tasks, dialect information and speaker identity do not significantly help performance, and hence are omitted in the presented experiments. Gender information is used, and improves performance by up to 10% relative. 298 classes are used for the TIDigits task and 1078 for RM.

The usefulness of the locally weighted distance measure is shown in table 1. Results are presented for two different feature sets. The "cepstra" are sine-liftered cepstra and their first time derivatives. "Mida" is an improved variant of LDA based on Mutual Information [4]. It takes 24 mel-scale filterbank coefficients and its first and second time derivatives as input to produce 25 features for the RM task and 15 features for the TIDigits task. The recognition results for both preprocessings show a strong improvement over the uniformly scaled metric.

While the results of this simple implementation are satisfying, it should be stressed that there is still a lot of room for improvement. Discriminative weight estimation and class assignment as well as the use of full covariance matrices are obvious candidates for further research.

3. A FAST K-NEAREST NEIGHBOURS ALGORITHM FOR SPEECH RECOGNITION

3.1. Introduction

Our example-based approach to large vocabulary recognition uses a bottom-up template selector to heuristically limit the search space of the decoder [2]. The selector takes as input a number of nearest neighbours and investigates their time evolution to detect interesting templates. As we want to use the complete training database

during recognition, a full distance calculation to all database vectors for each input frame would be too time-consuming. Hence a fast k-nearest neighbours (kNN) algorithm is essential to keep the computational complexity of our example-based recognizer within reasonable bounds. Although there is ample choice of different kNN algorithms in literature, our current problem deviates considerably from mainstream kNN research.

- The combination of a moderate dimensionality and a large number of vectors poses problems for typical algorithms such as the kd-tree [5] that perform very well for small dimensionality. This problem is widely known as the curse of dimensionality.
- The number of requested nearest neighbours k is a few orders of magnitude larger than in typical applications of kNN. In our recognizer, k will be somewhere between 10^3 and 10^5 depending on database size.
- Successive queries are samples from a continuous speech signal, and hence they will be correlated and evolve gradually through the acoustic space. Therefore, the solution of one query can conceivably be a valuable starting point for a successive query in an iterative algorithm.

But the main problem for fast kNN algorithms is posed by the non-uniform distance measure. Efficient hierarchical methods are based on geometrical properties, requiring a uniform distance metric or complex adaptations to compensate for varying distance measures. These adaptations are not feasible for large values of k. For a class-based distance measure, building a tree for each of the M classes after suitable scaling and then merging M sets of k nearest neighbours is an obvious solution. However, we want to use fine-grained classes or even a smoothly varying distance measure, where each point in space can give rise to a different scaling. In this case, hierarchical algorithms are infeasible.

Because correct kNN searches have turned out to be too computationally expensive, approximate kNN algorithms have received a lot of attention [6]. In our case, errors in the selected nearest neighbours can be recovered on a higher level by the time filter [2]. Hence we are satisfied with an approximate kNN algorithm.

3.2. Extended Roadmap algorithm

Our solution is an extended version of the Roadmap algorithm [7]. It is a *hill-climbing* search through a neighbourhood graph (or "Roadmap"). All arcs are bidirectional, and each acoustic database vector is a vertex. The general idea is that vertices that are *similar* are linked. When looking for nearest neighbours of a query vector, the current solution is iteratively improved by moving to an adjacent vertex that is closer to the query vector.

However, since two vectors that are very similar in acoustic space can belong to classes with different associated distance measures, *similarity* has to take into account both the vector itself and its associated distance measure. While the algorithm can cope with a smoothly varying distance measure, strong differences would make the error surface too complicated. Therefore, we use subgraphs for vectors having distinct associated distance measures. For example, for the experiments in this paper, all vectors labeled by the same HMM state number were taken together in one subgraph. Hence, for major differences in distance measure (different HMM-state) a new subgraph was formed, while small differences (different gender) within a subgraph were allowed.

Both the training algorithm and the search differ considerably from the original algorithm [7], which was intended as a fast selection of Gaussians in a Gaussian mixture HMM system.

3.2.1. The training algorithm

Because of the properties of our training algorithm, it is sufficient to create the different subgraphs during the initialization, e.g. by cyclic linking. For the rest of the training, the partitioning is of no influence on the algorithm's time complexity.

At each training iteration, three basic optimizations are performed on each database vector in pseudo-random order:

1. A small number of nearest neighbours in its subgraph (we used 8 in the experiments) are found for the vector, and those are linked to it using the linking procedure described below.
2. Another small number of randomly chosen vectors from the same subgraph is linked using the same procedure.
3. The Roadmap is "cleaned" by removing redundant arcs. Redundancy is also defined below.

One main difference with the original training algorithm is that the number of neighbours per vector is not fixed. While it makes basic operations more complicated, it has the definite advantage of allowing us to keep the Roadmap partitioning fixed. In the original algorithm, training could further break up graphs in uncontrolled fashion. A disadvantage of using a variable number of arcs per vertex is that the number of arcs might keep growing or might stay too small. Hence in our training algorithm the average number of arcs per vertex is controlled by adapting various parameters.

Linking is not done by simply adding a bidirectional arc. Rather, a fast constrained greedy search is used to get as close as possible to the target vertex starting from the source vertex. If the target vector can be reached, no arc is added. Otherwise, another greedy search is performed, this time starting from the target vector, trying to reach the endpoint of the first greedy search. An arc is added between the two endpoints. The greedy searches are constrained to consist of only a small number of steps, have a very small backtracking depth and are not allowed to increase the distance to the target with more than a given factor (in fact this factor can be smaller than one, hence constraining the search to only take steps in the correct direction). The idea of linking in this way is that the Roadmap is improved in function of the search, rather than just in function of obtaining better locality.

Cleaning of the Roadmap is done by checking for each arc if, without it, its connected vertices can still reach each other using the same kind of constrained greedy search. If so, it is deleted. This type of cleaning guarantees that the initial graph partitioning is maintained.

3.2.2. The search

The search consists of two separate parts.

- In the first part, a greedy search is performed for each of the subgraphs. The result is a small number of nearby vectors from each of the subgraphs. The backtracking depth of this search is small, causing very fast termination, but more risk of errors. The initialization is given by the result of the greedy search for the previous input frame. Often, that initial solution will hardly need to be changed. The time complexity of this phase is hard to calculate exactly.

Given some assumptions about the shape of the Roadmap, and given the worst possible initialization for each query, it is $O(MD^3 \sqrt[D]{n/M})$, with M the number of subgraphs, and D the dimensionality. In practice, the first phase is computationally negligible compared to the second, especially because the initialization will often be near-optimal.

- In the second phase, the best vectors found in the first phase are expanded to find k near neighbours. As in [7], we make extensive use of hashtables to avoid calculating distances more than once, but we also use a more efficient data structure consisting of an extra hashtable and two connected heaps to store the k current best vertices. This way the time complexity is limited to $O(Dklog(k))$. Note that the second phase is no longer performed on each of the subgraphs separately, thus avoiding extra calculations necessary to merge M sets of k candidate nearest neighbours.

The dominating second phase of the search is independent of n. This property is very desirable from a theoretical point of view, but in the practical case of our example-based speech recognizer, using larger databases will mean calculating more neighbours, too. We have not yet investigated the correct dependence of k on n, but our experience tells us that it is definitely sub-linear.

3.3. Scaling experiments

While worst-case asymptotic behaviour is important for a theoretical assessment of search algorithms, they can hide large constant factors that dominate in practical finite-size experiments. Furthermore, the presented algorithm is an *approximate* kNN algorithm, and it is easy to imagine that there is a trade-off between training time, search error rate and search time. Therefore, we present a realistic set of experiments to show the practical scaling behaviour of the algorithm. All experiments are given on TIDigits acoustic data, with the locally weighted distance measure that produced the best result in table 1. The effect of the number of training iterations is not examined, and kept constant at a level that takes approximately 1 CPU-day on a modern PC for the full training database.

In the first scaling experiment (top of fig. 1) the complete training database is used, and the number of requested nearest neighbours k is scaled from 1000 to 25000. For the recognition experiments in section 2.3, 15000 nearest neighbours were used. The left vertical axis combined with the full line show that scaling behaviour is linear in k. The values on that axis are the mean execution times per query on a 1.7 GHz CPU. The right axis with the dash-dotted line shows the percentage of nearest neighbours that weren't found. These misses occur mainly for the most distant of all neighbours. In all observed cases, the best neighbour was found.

In the second experiment (bottom of fig. 1) k is kept constant at 10000, while the database is subsampled randomly and scaled from about 100.000 frames to the full 1.516.015. The number of calculated distances (which is exactly linear with execution time) is shown to increase less and less with increasing database size.

4. CONCLUSIONS

We extended our example-based large vocabulary speech recognizer with a more powerful distance measure. The relationship with HMM modeling was clarified. A simple-proof-of-concept experiment showed the need for non-uniform scaling and the validity of the new distance measure. In all experimental set-ups, relative

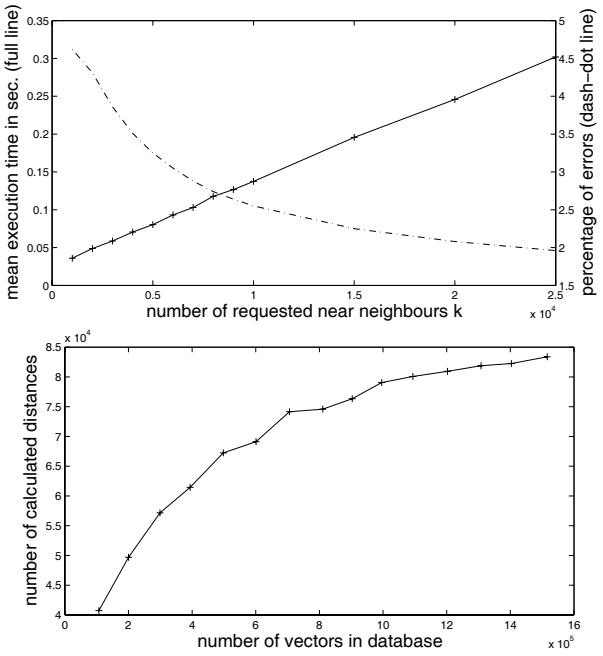

Fig. 1. *scaling k (top) and n (bottom)*

improvements in word error rate of more than 35% were obtained. Also, we addressed algorithmic problems that are introduced by the new distance measure. An extended version of the Roadmap algorithm was outlined and shown to perform well with increasing problem size.

5. REFERENCES

[1] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Trans. on ASSP*, vol. 23, no. 1, pp. 67–72, February 1975.

[2] M. De Wachter, K. Demuynck, D. Van Compernolle, and P. Wambacq, "Data driven example based continuous speech recognition," in *Proc. EUROSPEECH*, Geneva, Switzerland, Sept. 2003, pp. 1133–1136.

[3] J.J. Odell, *The Use of Context in Large Vocabulary Speech Recognition*, Ph.D. thesis, University of Cambridge, U.K., 1995.

[4] K. Demuynck, J. Duchateau, and D. Van Compernolle, "Optimal feature sub-space selection based on discriminant analysis," in *Proc. EUROSPEECH*, Budapest, Hungary, Sept. 1999, vol. III, pp. 1311–1314.

[5] J. R. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software*, vol. 3, no. 3, pp. 209–226, September 1977.

[6] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions," *Journal of the ACM*, vol. 45, no. 6, pp. 891–923, November 1998.

[7] D. Povey and P. C. Woodland, "Frame discrimination training of HMMS for large vocabulary speech recognition," Tech. Rep. CUED/F-INFENG/TR332, Cambridge University Engineering Department, 2000.