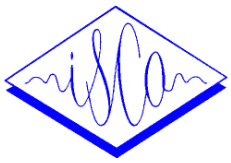

3rd International Conference on
Spoken Language Processing (ICSLP 94)
Yokohama, Japan
September 18-22, 1994

Fast Log-Likelihood Computation for Mixture Densities in a High-Dimensional Feature Space

Peter Beyerlein
Philips GmbH Forschungslaboratorium Aachen,P.O.Box 1980, D-52021 Aachen, Germany

Abstract

A computationally very expensive task arising within speech recognition systems using continuous mixture density HMMs is the log-likelihood computation. In the Philips large vocabulary continuous-speech recognition system it consumes 50% - 75% of the computing resources.
In our system the log-likelihood computation amounts to a nearest-neighbor search, i.e. to a search for the component density of a mixture density whose mean vector has a minimal distance to the observed feature vector.
Fast nearest-neighbor search techniques based on the triangle inequality are very powerful if the dimension of the feature space is lower than about 10. However, a direct application of these techniques is prohibitive in our framework which is characterized by a high-dimensional feature space and a small number of component densities per mixture density. In a typical setup we have 120 component densities per mixture density and a dimension of 63.
This paper

- introduces an efficient nearest-neighbor search algorithm adapted to the conditions of a high dimensional feature space and sparse data,
- gives a theoretical explanation and an experimental validation for the constraints of fast nearest-neighbor search techniques in a high-dimensional space.

1 Introduction

The acoustic modeling in the Philips large vocabulary continuous-speech recognition system [3],[4] is based on continuous mixture density hidden Markov models.
The likelihood of the observation vector $\vec{o}$ is given by

$$p(\vec{o}) = \sum_{k=1}^{K} w(k) \cdot p(\vec{o}|k),$$

where $w(k)$ is the weight of the k-th Laplacian mixture density component. After scaling we arrive at component densities with a pooled, scalar variance. Replacing the sum of probabilities by the maximum and taking the logarithm we obtain the nearest-neighbor rule for the log-likelihood $\log(p(\vec{o}))$:

$$\log(p(\vec{o})) \approx A - \min\{d(\vec{x}, \vec{a}(k)) - \log w(k) | k = 1, ..., K\},$$

where A is some constant and $d(\vec{x}, \vec{a}(k))$ is the city-block distance (L_1-Norm) between scaled observation vector $\vec{x}$ and scaled location vector $\vec{a}(k)$. We now define

$$\begin{aligned} \vec{y}^T &= (\vec{x}^T, 0) \\ \vec{r}(k)^T &= (\vec{a}(k)^T, -\log w(k)), \end{aligned}$$

and we denote $\vec{r}(k)$ as prototype k. Then we get

$$\begin{aligned} l(\vec{x}) &= \min\{d(\vec{y}, \vec{r}(k)) : k = 1, ..., K\} \\ d(\vec{y}, \vec{r}(k)) &= \sum_{c=1}^{D+1} |y_c - r_c(k)|. \end{aligned}$$

The task is now to find the prototype k nearest to the observed feature vector $\vec{y}$ in an efficient way.

2 Description of Distance Computation Algorithms

2.1 The Recall-Jump-Eliminate-Algorithm

We first describe the triangle-inequality-*elimination* technique [1]. Let $\vec{x}, \vec{y}, \vec{z}$ be $(D+1)$-dimensional vectors in a metrical feature space with distance function $d(\vec{x}, \vec{y})$. The triangle inequality can be written as $|d(\vec{x}, \vec{y}) - d(\vec{y}, \vec{z})| \leq d(\vec{x}, \vec{z})$. Let $d(\vec{x}, s)$ be the distance between the observation $\vec{x}$ and prototype s, $d(i, s)$ be the distance between prototypes s and i, and d_{min} the current nearest-neighbor distance. Assume all $d(i, s)$ are known. If $|d(\vec{x}, s) - d(i, s)| > d_{min}$ holds, $|d(\vec{x}, i)| > d_{min}$, and prototype i cannot be the nearest neighbor, i.e. it can be eliminated without distance computation. To reduce the overhead produced by unsuccessful elimination trials the following technique is introduced.

1. **Fast elimination**
Within each HMM-state do: For each prototype r compute a list L_r containing the distances to all the other prototypes, sorted by ascending distance. After the distance computation of the prototype i to the observation $\vec{x}$, update the minimum distance d_{min} and compute the two distance bounds $d_H = d(\vec{x}, i) + d_{min}$ and $d_L = d(\vec{x}, i) - d_{min}$. Then search on the list L_i of prototype i for the first prototype j_L with $d(j_L, i) \geq d_L$ and for the last prototype j_H with $d(j_H, i) \leq d_H$. As only list elements between j_L and j_H are possible nearest-neighbor hypotheses, eliminate the others.
The search strategy of the algorithm is as follows:

2. **Recall-and-Jump**
The search for the nearest neighbor to the observation $\vec{x}_n$ (n denotes a time index) starts with the nearest neighbor of the previous observation $\vec{x}_{n-1}$ ("Recall").
In a preprocessing step the prototypes are ordered according to their pairwise distance. For each prototype the maximum distance d_m to all the other prototypes is computed. The prototypes are ordered according to d_m. The search starts with those prototypes that have a maximal d_m, i.e., a maximal pairwise distance ("Jump").

The above described algorithm was introduced in [6] and

	runtime R	word error rate	looseness L
FULL	1.00	22.82	0.0 (exact)
PD	0.83	22.82	0.0 (exact)
RJE	0.68	22.82	0.0 (exact)
RJE	0.60	22.87	0.2
RJE	0.52	22.70	0.4
RJE	0.43	23.80	0.6

Table 1: Results for N=30 with
FULL (straightforward exhaustive full search),
PD (Partial-Distance Algorithm) and
RJE (Recall Jump Eliminate Algorithm)

is called RJE (Recall-Jump-Eliminate).

3. **Looseness**
To further increase the number of eliminated prototypes the triangle inequality can be tightened by a so-called 'looseness'-parameter l:
If $|d(\vec{x},s) - d(i,s)| > d_{min} - l$ holds and l is close to zero, one would expect that prototype i is not the nearest neighbor, i.e. it will be eliminated without distance computation.
This leads to a suboptimal but well performing version of the RJE.

2.2 The Partial-Distance Algorithm

The partial distance of two vectors is defined as the sum of the first c contributions ($1 \leq c \leq D+1$) of the vector components to the distance of the two vectors.
In testing a new prototype, the prototype is rejected without completing the distance calculation if the partial distance exceeds the current minimum distance [5].
The search for the nearest neighbor to the observation $\vec{x}_n$ starts with the nearest neighbor of the previous observation $\vec{x}_{n-1}$.

3 Experimental Results

The aim of the application of the previously described algorithms was to improve the nearest-neighbor search in our baseline system. The test set consists of 30 min of speech material. The feature space dimension is $D = 63$. In the first setup there are about $N = 30$ prototypes in each of the 127 HMM-states. In the second setup there are about $N = 120$ prototypes in each of the 127 HMM-states. Results for different looseness-values l are given in table 1 for $N = 30$ and in table 2 for $N = 120$, were R and L are defined as:

$$R = \frac{\text{runtime of the nearest-neighbor search}}{\text{runtime of an exhaustive full nearest-neighbor search}}$$

$$L = \frac{\text{looseness value l (to tighten the triangle inequality)}}{\text{mean distance of training observations and prototypes}}.$$

The results show that we could find a well performing but suboptimal nearest-neighbor search for our setup. However, it is important to know the constraints of a fast nearest-neighbor search under certain conditions. Therefore the next section shows how the dimension and number of prototypes influence the performance of the nearest-neighbor search.

	runtime R	word error rate	looseness L
FULL	1.00	14.67	0.0 (exact)
PD	0.83	14.67	0.0 (exact)
RJE	0.58	14.67	0.0 (exact)
RJE	0.49	14.69	0.2
RJE	0.36	15.04	0.4
RJE	0.32	15.68	0.6

Table 2: Results for N=120

4 On the Theoretical Bounds of a Fast Nearest-Neighbor Search

To show how the dimension and number of prototypes influence the performance of a nearest-neighbor search we derive an expression of the m-th order moment $\mathbf{E}(\frac{d_1}{d_N})^m$ of the distance ratio of the nearest-neighbor distance d_1 and the farthest-neighbor distance d_N to the observation vector.
Let $\vec{a_1}, ..., \vec{a_N}$ be independent and identically distributed prototype vectors in a D-dimensional real-valued feature space $\mathcal{R}^D$ with probability density function $h(\vec{a})$. Let $B(\vec{x},r)$ be a Ball in $\mathcal{R}^D$ with radius r, i.e. $B(\vec{x},r) = \{\vec{y}\epsilon\mathcal{R}^D : \|\vec{y}-\vec{x}\| \leq r\}$, where $\|.\|$ is a given norm. Now we define $C(\vec{x},r) = \int_{B(\vec{x},r)} h(\vec{y})d\vec{y}$. We call $C(\vec{x},r)$ the **coverage** of the region $B(\vec{x},r)$. $C(\vec{x},r)$ is the probability that a prototype $\vec{a}$ will fall into the closed Ball $B(\vec{x},r)$. In the following we assume that the $\vec{a_i}$ are sorted by $d_1 \leq d_2 \leq ... \leq d_N$, with $d_i = \|\vec{a_i} - \vec{x}\|$. Since the $\vec{a_i}$ are random vectors, the $C_k = C(\vec{x},d_k)$ are random values. Based on [7] it can be shown that their joint probability density function is given by

$$f_c(C_1,...,C_k) = \frac{N!}{(N-k)!}(1-C_k)^{N-k}.$$

Now we compute the expected value $\mathbf{E}(\frac{C_l}{C_k})^\alpha$, l<k :
Let $\mathcal{C} = \{(C_1,...,C_k) \mid 0 \leq C_1 \leq C_2 \leq ... \leq C_k \leq 1\}$.
Then we can write

$$\mathbf{E}(\frac{C_l}{C_k})^\alpha = \int_{\mathcal{C}} (\frac{c_l}{c_k})^\alpha f_c(c_1,...,c_k)dc_1...dc_k.$$

After integration we obtain

$$\mathbf{E}(\frac{C_l}{C_k})^\alpha = \frac{\Gamma(l+\alpha)\Gamma(k)}{\Gamma(k+\alpha)\Gamma(l)} \qquad (1)$$

Note that $\mathbf{E}(\frac{C_l}{C_k})^\alpha$ is independent of N. Based on the expected coverage ratio in (1) we want to find an expression for the expected distance ratio $\mathbf{E}(\frac{d_l}{d_k})^\alpha$.
Let us assume that $h \epsilon C^2(\mathcal{R}^D, \mathcal{R})$. Then we can write

$$h(\vec{y}) = h(\vec{x}) + \nabla h(\vec{x})(\vec{y}-\vec{x}) + (\vec{y}-\vec{x})\left(\int_0^1\int_0^t H(\vec{x}+s[\vec{y}-\vec{x}])dsdt\right)(\vec{y}-\vec{x})^T,$$

were the elements of matrix H are defined by
$H_{ij}(\vec{x}) = \frac{\delta^2}{\delta x_i \delta x_j} h(\vec{x})$. Integrating over $B(\vec{x},r)$ we get

$$C(\vec{x},r) = V(\vec{x},r)h(\vec{x})(1+\Theta(\vec{x},r)) \quad \text{, with}$$

$$V(\vec{x},r) = \int_{B(\vec{x},r)} d\vec{y} \quad \text{, and}$$

$$\Theta(\vec{x},r) = \frac{\int_{B(\vec{x},d_k)}(\vec{y}-\vec{x})\left(\int_0^1\int_0^t H(\vec{x}+s[\vec{y}-\vec{x}])dsdt\right)(\vec{y}-\vec{x})^T d\vec{y}}{V(\vec{x},r)h(\vec{x})}.$$

So,

$$\frac{C_l}{C_k} = \frac{V(\vec{x}, d_l)(1 + \Theta(\vec{x}, d_l))}{V(\vec{x}, d_k)(1 + \Theta(\vec{x}, d_k))}. \quad (2)$$

Since $V(\vec{x}, r) = K_D \cdot r^D$ holds [1], we find

$$\frac{C_l}{C_k} = \frac{{d_l}^D(1 + \Theta(\vec{x}, d_l))}{{d_k}^D(1 + \Theta(\vec{x}, d_k))}. \quad (3)$$

4.1 Influence of Dimension and Number of Prototypes

The statistical structure of the data is fully included in the term $\Theta(\vec{x}, d)$. If the matrix H is small enough we have $\Theta(\vec{x}, d) \approx 0$ and the influence of the structure of the data vanishes. From (3) and (1) follows:

$$\mathbf{E}[(\frac{d_l}{d_k})^m] = \mathbf{E}[(\frac{C_l}{C_k})^{\frac{m}{D}}] \quad (4)$$

$$= \frac{\Gamma(l + \frac{m}{D})\Gamma(k)}{\Gamma(k + \frac{m}{D})\Gamma(l)}. \quad (5)$$

For mean and variance of the distance ratio $\frac{d_1}{d_N}$ of the nearest neighbor and the farthest neighbor to $\vec{x}$ we obtain:

$$\mathbf{E}[\frac{d_1}{d_N}] = \frac{\Gamma(1 + \frac{1}{D})\Gamma(N)}{\Gamma(N + \frac{1}{D})} \quad (6)$$

$$\mathbf{Var}[\frac{d_1}{d_N}] = \frac{\Gamma(1 + \frac{2}{D})\Gamma(N)}{\Gamma(N + \frac{2}{D})} - \left(\frac{\Gamma(1 + \frac{1}{D})\Gamma(N)}{\Gamma(N + \frac{1}{D})}\right)^2 \quad (7)$$

The most important conclusions can be derived for $D \gg 1$. In this case we can approximate

$$\mathbf{E}[\frac{d_1}{d_N}] \approx \left(\frac{1}{N}\right)^{\frac{1}{D}} \quad (8)$$

$$\mathbf{Var}[\frac{d_1}{d_N}] \approx 0 \ , \quad (9)$$

i.e. the N prototypes will fall in a thin shell around the point $\vec{x}$. The ratio of the inner shell-radius and the outer shell-radius is given by $(\frac{1}{N})^{\frac{1}{D}}$. Figure 4 shows a plot of (8) with $e(N, D) = \mathbf{E}[\frac{d_1}{d_N}]$ for different values of dimension D and number of prototypes N.

4.2 Discussion and Experimental Validation

4.2.1 Discussion

The search effort for a nearest-neighbor search depends directly on the distance ratio $\frac{d_1}{d_N}$. If this ratio is small we find prototypes near to the observation $\vec{x}$ and prototypes far from the observation. Hence, the nearest-neighbor search can perform well. However, if $\frac{d_1}{d_N} \approx 1$, almost all prototypes can be found on a sphere around the observation vector $\vec{x}$. In this case all distances had to be computed to find the nearest neighbor. Given the assumptions in the previous section and the derived equations (6) and (7) we can draw following conclusions:

[1] K_d is some konstant

component	feature	feature region
1 .. 30	spectral intensities I	static
31	energy W	static
32 .. 46	first derivatives of I	dynamic
47	first derivative of W	dynamic
48 .. 62	second derivatives of I	dynamic
63	second derivative of W	dynamic

Table 3: Feature vector

dimension D	static region	dynamic region
2	8%	12%
4	9%	15%
8	11%	25%
16	17%	42%
32	24%	53%

Table 4: Rate E of computed distances versus dimension and feature region of the feature vectors

- In a setup with large N and small D ($D < 10$) , e.g. vector quantization, nearest-neighbor search investigations are promising, since we have a moderate distance ratio.
- In a typical mixture-density setup with small N and large D we cannot find a universal efficient nearest-neighbor search algorithm. I.e., well-performing nearest-neighbor search techniques are expected to be strongly data-dependent.

The above equations were derived under the assumption that the term $\Theta(\vec{x}, r)$ vanishes. Since $\Theta(\vec{x}, r)$ includes all information about the probability density $h(\vec{x})$, the above equations (6) and (7) do not include any information about the real speech data. However the following experiments will show, that we find the described characteristic behaviour in the case of real data, too. What we want to show is

- that with increasing dimension D the number of distance computations increases
- that with increasing number N of prototypes the number of distance computations decreases.

4.2.2 Experimental Validation

In our setup the feature vector dimension is $D = 63$. Table 3 shows the structure of our feature vectors.

Define rate E as:

$$E = \frac{\text{number of computed distances}}{\text{number of prototypes}}.$$

The figures 1, 2 and 3 and table 4 show the behavior of the algorithms PD and RJE.

- Fig. 1 demonstrates that rate E increases monotonically with the feature space dimension.
- The influence of the number of prototypes within a HMM-state is shown in fig. 2. The rate E decreases with an increasing number of prototypes.
- In the next experiment (table 4) different feature subsets of the feature vectors were considered. The rate E as a function of dimension D has the same characteristic course in both feature regions (compare fig. 1, influence of dimension). In the dynamic feature region,

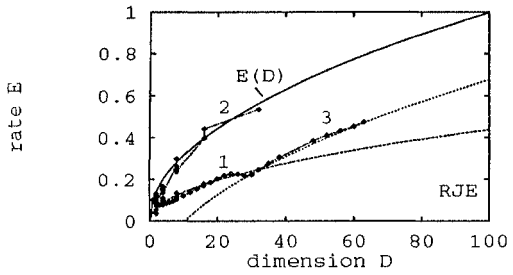

Figure 1: Rate E of noneliminated prototypes versus dimension
1 - static feature region
2 - dynamic feature region
3 - static + dynamic feature regions

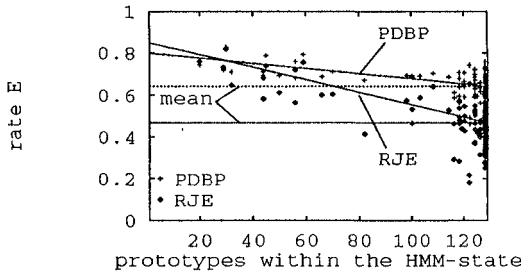

Figure 2: Rate E of noneliminated prototypes versus number of prototypes within the HMM-state

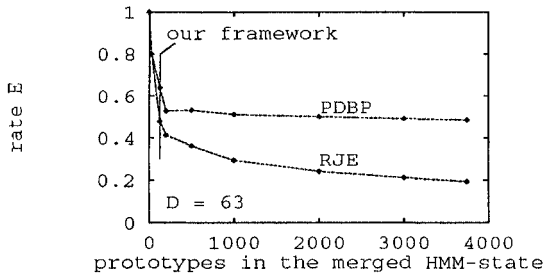

Figure 3: Rate E of computed distances versus number of prototypes within joined HMM-states

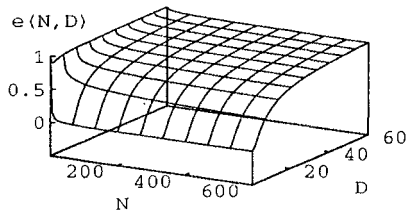

Figure 4: Distance ratio e(N,D) of the nearest neighbor and the farthest neighbor given by equation (8)

it increases more rapidly with the dimension. Hence, the influence of the term $\Theta(\vec{x}, r)$ is obvious, if we compare the two feature regions.

- The combined influence of the structure of the feature space and the number of prototypes is shown in fig. 3. During this experiment, an increasing number of prototypes of different HMM-states was merged into one set of prototypes.

5 Conclusion

Different approaches for a fast log-likelihood computation in a large vocabulary speech recognition system have been studied. The system is characterized by a small set of prototypes per HMM-state and a high-dimensional feature-space. The nearest-neighbor search runtime could be reduced by 50%-60% without introduction of additional recognition errors using a suboptimal version of our RJE-algorithm. The latter fits very well into the conditions of sparse data and high-dimensional feature-spaces.
The RJE-algorithm searches for the nearest neighbor only within the given mixture density. Hence, other techniques based on the whole density inventory, e.g. the vector quantization approach in [2], could be combined with the RJE to further decrease the computational resources required for the log-likelihood computation.
Using the theory of nonparametric density estimation, it was shown that with increasing dimension and decreasing number of prototypes the performance of an optimal nearest-neighbor search decreases. The influence of these factors on the number of distances that must be computed during the nearest-neighbor search was experimentally validated.

6 References

[1] Enrique Vidal Ruiz. An Algorithm For Finding Nearest Neighbors in (Approximately) Constant Average Time. Pattern Recognition Letters 4, 1986.
[2] E. Bocchieri. Vector Quantization for the Efficient Computation of Continuous Density Likelihoods. Proc. ICASSP, 1993. pp. 692-695
[3] H. Ney. Acoustic Modelling of Phoneme Units for Continuous Speech Recognition. Signal Processing V: Theory and Applications, 1990.
[4] H. Ney, V. Steinbiss, R. Haeb-Umbach, B.-H. Tran, U. Essen. An Overview of the Philips Research System for Large-Vocabulary Continuous-Speech Recognition. To appear in International Journal of Pattern Recognition and Artificial Intelligence, 1994.
[6] Peter Beyerlein. Experiments on a Fast Nearest-Neighbor Search for Mixture Density Likelihood Computation. To appear in Proc. NATO-ASI, New Advances and Trends in Speech Recognition and Coding, Bubion, 1993
[5] Jon Luis Bentley. Writing Efficient Programs. Prentice Hall International, London, 1982. pp. 67,68.
[7] K. Fukunaga. Introduction to Statistical Pattern Recognition, Second Edition. Academic Press, 1990. pp. 269,270.