UNIVERSITY OF JOENSUU
COMPUTER SCIENCE AND STATISTICS
DISSERTATIONS 22

Ville Hautamäki

Improving Pattern Recognition Methods for Speaker Recognition

Academic dissertation

To be presented, with the permission of the Faculty of Science of the University of Joensuu, for public criticism in the Louhela Auditorium of the Science Park, Länsikatu 15, Joensuu, on October 3th 2008, at 12 o'clock noon.

UNIVERSITY OF JOENSUU
2008

Supervisor Professor Pasi Fränti
Department of Computer Science and Statistics
University of Joensuu
Joensuu, FINLAND

Reviewers Professor Martti Juhola
Department of Computer Sciences
University of Tampere
Tampere, FINLAND

Professor Olli Nevalainen
Department of Information Technology
University of Turku
Turku, FINLAND

Opponent Professor Pekka Kilpeläinen
Department of Computer Science
University of Kuopio
Kuopio, FINLAND

ISBN 978-952-219-159-5 (printed)
ISBN 978-952-219-160-1 (PDF)
ISSN 1796-8100 (printed)
ISSN 1796-8119 (PDF)
Computing Reviews (1998) Classification: I.2.7, I.5.1, I.5.4, I.5.3, G.1.6, G.2.2

Joensuun yliopistopaino

Joensuu 2008

Improving Pattern Recognition Methods for Speaker Recognition

Ville Hautamäki

Department of Computer Science and Statistics
University of Joensuu
P.O.Box 111, FIN-80101 Joensuu, FINLAND
villeh@cs.joensuu.fi

University of Joensuu, Computer Science and Statistics, Dissertations 22
Joensuu, 2008, 126 pages

Abstract

AUTOMATIC speaker recognition is a very active area of research. The goal of speaker recognition is to either verify, based on voice only, that the user is who he claims to be or to identify unknown person from the voice sample. However, given the potential of the speaker recognition technology itself, it is still not widely used in commercial applications. The reason is that the speaker recognition is still an immature technology, where baseline technology is not very robust to various mismatches between training and testing conditions.

Currently, improvements to speaker recognition methodology are aimed either to pattern recognition techniques or signal processing and feature extraction procedures. In this thesis, the focus is mainly in the pattern recognition part.

Cluster analysis is the basic technology utilized in the speaker modeling. The first three publications improve the clustering results by speeding up the agglomerative clustering algorithm, developing an outlier removal technique, and creating a robust variant to the cluster centroid estimation.

A vector quantization based maximum *a posteriori* speaker model adaptation is also formulated. The adaptation strategy increases the accuracy of the vector quantization system to the same level as the state-of-the-art GMM-UBM system, while being 20 times faster in speaker training.

Mismatch compensation in speaker recogntion can be done for example by techniques from factor analysis. These recent techniques require external speech data, where statistical techniques will estimate the nuisance attributes. In this thesis, we propose a novel speaker matching and modeling method. The method is based on

graph matching and does not use any external training data.

Finally, we also propose a new voice activity detector for speaker recognition. We also systematically try out several speaker matching and feature extraction methods for the long-term average spectrum feature.

Keywords: Text-independent speaker recognition, vector quantization, cluster analysis, outlier detection, affine invariant matching, maximum a posteriori adaptation, unsupervised learning

Acknowledgements

I would like to express my sincere thanks to my supervisor Professor Pasi Fränti. I am especially grateful on the extremely successful collaborative work achieved with Dr. Tomi Kinnunen and Dr. Ismo Kärkkäinen. Other co-authors of the papers included in this thesis also deserve thanks, they are Dr. Olli Virmajoki, Svetlana Cherednichenko, Marko Tuononen, Juhani Saastamoinen and Tuija Niemi-Laitinen. Special thanks also goes for the proof reading done by Lomanzi Sakala and Pekka Nykänen.

I would also like to thank Professor Olli Nevalainen and Professor Martti Juhola for their effort on the review of this thesis. I am especially grateful for the helpful language editing provided by the above mentioned reviewers.

I would like to thank for the financial support East Finland Graduate School in Computer Science and Engineering (ECSE) during the years 2007-2008. The work on the thesis was also supported by the National Technology Agency of Finland (TEKES) as the four year project New Methods and Applications of Speech Technology (PUMS), and by all the government organizations and private companies participating in this project.

My parents and grand-parents have supported me tremendously during the long years of acquiring an education, for this and everything else they are greatly thanked for. Finally, I am forever indebted to my dear wife Rosa and our twin boys Diego and Samuel, they are the love of my life.

Joensuu, Tuesday, 9 September 2008

Ville Hautamäki

Let your vision be world-embracing
Bahá'u'lláh (1817–1892)

List of original publications

P1. P. Fränti, O. Virmajoki and V. Hautamäki, "Fast agglomerative clustering using a k-nearest neighbor graph", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, No. 11, pp. 1875–1881, November 2006.

P2. V. Hautamäki, I. Kärkkäinen and P. Fränti, "Outlier detection using k-nearest neighbour graph", in *Proceedings of the 17th International Conference on Pattern Recognition* (ICPR 2004), Vol. 3, pp. 430–433, Cambridge, UK, August 2004.

P3. V. Hautamäki, S. Cherednichenko, I. Kärkkäinen, T. Kinnunen and P. Fränti, "Improving k-means by outlier removal", in *Proceedings of the 14th Scandinavian Conference on Image Analysis* (SCIA 2005), pp. 978–987, Joensuu, Finland, June 2005.

P4. V. Hautamäki, T. Kinnunen, I. Kärkkäinen, J. Saastamoinen, M. Tuononen and P. Fränti, "Maximum *a posteriori* adaptation of the centroid model for speaker verification", *IEEE Signal Processing Letters*, Vol. 15, pp. 162–165, 2008.

P5. V. Hautamäki, T. Kinnunen and P. Fränti, "Text-independent speaker recognition using graph matching", *Pattern Recognition Letters*, Vol. 29, No. 9, pp. 1427–1432, July 2008.

P6. T. Kinnunen, V. Hautamäki and P. Fränti, "On the use of long-term average spectrum in automatic speaker recognition", in *Proceedings of the 5th International Symposium on Chinese Spoken Language Processing* (ISCSLP 2006), Vol. 2, pp. 559–567, Singapore, December 2006.

P7. V. Hautamäki, M. Tuononen, T. Niemi-Laitinen and P. Fränti, "Improving speaker verification by periodicity based voice activity detection", in *Proceedings of the 12th International Conference on Speech and Computer* (SPECOM 2007), Vol. 2, pp. 645–650, Moscow, October 2007.

Contents

1 Introduction **1**

2 Clustering **3**

2.1 Proximity measures 4
2.2 Squared error criterion 5
2.3 Other criteria 7
2.4 k-means algorithm 9
2.5 Exact and approximation algorithms 11
2.6 Agglomerative algorithms 12
2.7 Graph clustering 13
2.8 Gaussian mixture modeling 14
2.9 Outlier detection 18

3 Speaker Recognition **21**

3.1 Front-end processing 22
3.2 Speaker modeling 25
3.3 Mismatch compensation 26

4 Summary of the Publications **29**

5 Summary of the Results **33**

6 Conclusions **37**

References **39**

Chapter 1

Introduction

BIOMETRIC person recognition from voice is a highly active research field [11]. The voice biometric has some advantages over more common biometric technologies, such as *fingerprint*, *iris* and *DNA* recognition. A person can be authenticated by his voice without him noticing it, as in border control usage scenario, where voice biometric system can run in the background while an officer is interviewing the passenger. Voice biometric system can also perform *realtime-recognition* [101, 113], for example, in call center operations, a user can be recognized while he is interacting with the system. The change of a user in the middle of the call can be detected and also more speech material can be collected during the whole call. *Face recognition* possesses similar properties but it is restricted to the operating situations where there is visual line of sight to the user.

Unfortunately, speech biometric is also inherently more difficult than more traditional means of person authentication using physical characteristics. Speech is a highly variable phenomenon. Variability is mostly due to the following aspects: (1) *speaker himself* (mood and health differences), (2) *technical conditions* (changes in environmental acoustics, transmission line and microphones, microphone setups) (3) *linguistic factors* (speech content, language, dialect and situation variations) [98]. The challenge in designing an automatic speaker recognition system is to make a system robust against the above mentioned variability and still retaining a good speaker discrimination ability.

An automatic speaker recognition system consists of the following components: voice activity detector (VAD), feature extraction, speaker modeling, pattern matching and speaker database. An overall system diagram is presented in Fig. 1.1. The speaker recognition task implemented in a complete system, is either *verification* or *identification*. In verification task, system is given an uknown speech sample and claimed speaker identity, the task is to decide whether unknown speech sample was

uttered by the claimed speaker. In the identification task, system will give speaker label to the unknown speech sample. Finally, the decision logic of the system returns the best scoring speaker or decision that the speaker is unknown.

Figure 1.1: Structure of a speaker recognition system.

Training a new speaker to the speaker database and matching an unknown audio extract to the claimed model are basic operations in all speaker recognition systems. Typically, feature extraction produces a so called "feature cloud", which needs to be modeled somehow. Statistical techniques are commonly used in the modeling part of the system and data clustering is an essential tool therein.

The rest of the thesis is organized as follows. In Chapter 2, clustering is considered as an optimization problem, its computational complexity is analyzed and algorithmic solutions are reviewed. Gaussian mixture models and outlier detection methods are also discussed in the same chapter. Chapter 3 reviews the speaker recognition literature and shows how the clustering is applied in the speaker modeling. Summary of original research papers is given in Chapter 4 and the main results are summarized in Chapter 5. Finally, conclusions are drawn and future work outlined in Chapter 6. The original research papers are attached at the end of the thesis.

Chapter 2

Clustering

ONE can state that the intuitive goal of clustering is to find *natural clusters* from the input data without using any additional information [83, 185]. In machine learning, this type of learning problem is also known as *unsupervised learning* in contrast to *supervised learning* where additional class labels are provided with the data. The lack of additional information means that in order to be able to solve the *clustering problem*, one has to assume a clustering model. The algorithmic problem can be stated as; given a data set, find a set of model parameters that minimize (or maximize) a given objective function. If the data set includes class labels, the clustering algorithm can be objectively evaluated by checking for each pair of data vectors, whether they have been assigned to the same cluster or not [149]. Unfortunately, there is no way to know whether the results obtained with labeled training data will generalize to the unsupervised case.

Clustering is performed on the input data set (or *training set*) $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ of N vectors in $\mathbb{R}^D$. The task is to partition X into k disjoint subsets (or *partitions*) S_i such that $\bigcup_{i=1}^{k} S_i = X$ and $S_i \cap S_j = \emptyset$ for all $j \neq i$. In the above definitions, each observation belongs to only one cluster. It is sometimes called *crisp* or *hard* clustering to distinguish from *soft* clustering definitions where an observation belongs to certain clusters by some membership degrees. There are number of soft clustering definitions including for example *fuzzy* [110], *probabilistic* [37] and *possibilistic* [109] clusterings.

Intuitively, each cluster should be internally as homogeneous as possible and, on the other hand, between-cluster similarity should be minimized [185]. Clustering, as defined above, does not include any *hierarchy* of clusters. Some clustering algorithms produce a successive hierarchy of clusters as a by-product. In the hierarchy, a child cluster is totally contained in its parent. Such a clustering can be represented by a *dendrogram*.

In the following sections, we will formalize the notion of cluster homogeneity. It uses the concept of *inter-observation dissimilarity measure* $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ or a *proximity measure* and the definition of the *clustering optimization criterion* [66]. The criterion will assign *cost* $f : (S_1, \ldots, S_k) \to \mathbb{R}$ for each partitioning of the training set.

2.1 Proximity measures

Intuitively, if a group of observations are close to each other and far away from the other observations, then they should form a cluster. Now, we need a formal definition of "closeness". In the case of clustering, a *dissimilarity measure* $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a *metric* [29] if it fulfills the following axioms:

1. $d(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$ (*non-negativity*),
2. $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = 0$, if and only if $\boldsymbol{x}_i = \boldsymbol{x}_j$ (*identity of indiscernible*),
3. $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = d(\boldsymbol{x}_j, \boldsymbol{x}_i)$ (*symmetry*) and
4. $d(\boldsymbol{x}_i, \boldsymbol{x}_k) \leq d(\boldsymbol{x}_i, \boldsymbol{x}_j) + d(\boldsymbol{x}_j, \boldsymbol{x}_k)$ (*triangle inequality*).

In clustering, triangle inequality is usually required, such a dissimilarity measures is called a *semimetric* [164]. In the rest of the thesis, we will refer to any dissimilarity function as a *distance function* and simply a *distance*.

All pairwise distances of the data set can be given a graph theoretic interpretation. Distances can be seen to form a distance matrix A, where elements a_{ij} are distances between data objects $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. It follows immediately from axiom 3 of metric spaces that A is a symmetric matrix. It can be interpreted as an *adjacency matrix* of a weighted complete undirected graph $G = (V, E)$. Here, V is the set of input vectors and E is the set of all $\binom{N}{2}$ edges. In this way, we can interpret the clustering problem as a *graph clustering* problem [157]. However, by analyzing the distance matrix, we can see that typically some vector pairs can never be in the same cluster. For that reason, as a preprocessing step, large distances are sometimes set to infinity and thus eliminated them from further processing [65].

Above mentioned simple thresholding scheme is one possibility, a more complicated scheme will compute a new *subgraph* from the original G. Subgraphs used in the literature are for example *minimum spanning tree* (MST) [30] and *k-nearest neighbour graph* (kNNG) [46]. A k-nearest neighbour graph is an undirected graph, where an edge is retained between vertices a and b if either b is one of the k-nearest neighbours of a or vise versa. An example of 1NNG in Euclidean plane is shown in Fig. 2.1. From a complete graph, a kNNG can be trivially computed in $O(N^2)$ time. In Euclidean space, it is possible to compute kNNG in $O(N \log N)$ time, but

with constants that are exponential with respect to the dimensionality of the data set [20, 28, 174]. It is an open problem whether a fast variant without large hidden constants is possible.

Figure 2.1: Directed 1NNG (left) and the undirected variant (right).

A commonly used metric between two vectors in a D-dimensional vector space is the l_p-norm or *Minkowski metric* [40]:

$$l_p(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{D} |x_i - y_i|^p \right)^{1/p} = \|\boldsymbol{x} - \boldsymbol{y}\|_p. \tag{2.1}$$

The most common special cases are l_1-, l_2- and l_∞-norms. The l_2-norm is also known as the *Euclidean distance*. From the Minkowski metric it is natural to define the proximity measure for the clustering problem, which we call *squared error*:

$$d(\boldsymbol{x}, \boldsymbol{y}) := l_p(\boldsymbol{x}, \boldsymbol{y})^2 = \|\boldsymbol{x} - \boldsymbol{y}\|_p^2. \tag{2.2}$$

When p is set to two, we get the *squared Euclidean distance* or just squared error. Generalized variant of the squared error is used in robust statistics, where exponent 2 is replaced by a user selectable parameter α [192].

2.2 Squared error criterion

In a *centroid model*, each cluster $S_i\,((i = 1, 2, \ldots, k))$ is represented by a vector (called *centroid* or *prototype*) $\boldsymbol{c}_i \in \mathbb{R}^D$, where $C = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k\}$. The goal is to find such clustering where the error given by the following criterion is minimized:

$$\frac{1}{|X|} \sum_{\boldsymbol{x} \in X} \|\boldsymbol{x} - \boldsymbol{c}_j\|_2^\alpha, j = 1, \ldots, k. \tag{2.3}$$

Fortunately, the clustering criterion can be simplified in the case of the squared error criterion; the optimal prototype $\boldsymbol{c}$ for a given vector $\boldsymbol{x}_i$ is its nearest prototype. Criterion (2.3) can be written in a form which is commonly called the *mean squared error* (MSE) [60]:

$$\frac{1}{|X|} \sum_{\boldsymbol{x} \in X} \min_{\boldsymbol{c} \in C} \|\boldsymbol{x} - \boldsymbol{c}\|^2. \tag{2.4}$$

Given a particular partitioning, the mean vector is the optimal representative to all vectors in the same cluster. If on the other hand, we set α to one, then the optimal cluster prototype $\boldsymbol{c}$ is the *spatial median* element [192]. It is found to be more robust to outliers than the mean vector but its computation is more involved [192]. It can be computed as a solution to *transportation problem*. In the rest of the thesis, we fix p and α to 2.

On the other hand, the total squared error of a cluster can be represented by [161]:

$$\sum_{\boldsymbol{x} \in S_j} \|\boldsymbol{x} - \boldsymbol{c}\|^2 = \frac{1}{|S_j|} \sum_{\boldsymbol{x}_i, \boldsymbol{x}_h \in S_j} \|\boldsymbol{x}_i - \boldsymbol{x}_h\|^2, \tag{2.5}$$

where $|S_i|$ is the cardinality of the cluster S_i. In this way, the squared error clustering criterion can also be formulated as a graph clustering problem. However, in graph partitioning formulation, a cluster is represented by partitioning the labels of the observations. The squared error criterion gives another possibility to represent the solution. In particularly, it can be represented by the centroid set C. Conveniently, in the model parameter estimation setting, we can form a long *parameter vector* $\Theta = (\boldsymbol{c}_1^t, \boldsymbol{c}_2^t, \ldots, \boldsymbol{c}_k^t)^t$, as was defined in [**P4**]. A minor problem in the parameter vector interpretation is that each parameter has a fixed position in the Θ while the clustering remains the same even if their positions are shuffled [12].

2.2.1 Computational complexity

Globally and locally optimal solutions of the clustering problem posed in (2.3) are characterized by the *centroidal Voronoi diagram* (CVD) [39]. A *Voronoi tessellation* or diagram is a partition of the feature space into regions (*cells*), where each region is represented by one site (vector) called *generator*. The feature space is partitioned in such a way that each vector is mapped to its nearest generator. This mapping produces Voronoi cells that are polyhedra where each hyperplane is equidistant from two generators and the points of intersections are equidistant of at least three generators [8]. A centroidal Voronoi diagram is a structure where generators and region centroids coincide in each region. Example of the centroidal Voronoi diagram is shown in Fig. 2.2 on the right and for contrast also a Voronoi diagram that is not centroidal is shown on the left.

Figure 2.2: Two examples of Voronoi diagrams. A generators are marked by closed circles and region centroids by open circles. Random clustering with 2.46 MSE (left) and the corresponding CVD with 1.04 MSE (left).

Each centroidal Voronoi tessellation of the data set is a locally optimal solution, and the minimum of all locally optimal solutions is the globally optimal solution. Unfortunately, globally optimal solution for the clustering problem is not unique. For a given data set and number of clusters many different solutions might exist with the same minimum criterion value [60]. However, when the data set contains a non-overlapping clustering structure, optimal clustering is unique and one can calculate the bound on how close a given clustering is from the optimal one [131].

The number of all possible Voronoi diagrams with k generators can be shown to be $O(N^{kD})$ [82]. This result yields a polynomial time algorithm to a clustering problem when k is fixed. Enumerating all possible Voronoi diagrams turns out take $O(N^{(kD+1)})$ time [82]. On the other hand, when k is a part of the input, for example $k = f(N)$, clustering has been proved to be an NP-complete problem [38, 121]. However, when clusters are restricted to be of equal size, the problem becomes polynomial [35, 81]. Also when clustering in a 2-dimensional space, k being part of the input, the problem is NP-complete but if k is fixed, clustering is solvable in $O(N^{6k})$ time [24]. It is clear, in which ever way we view the problem, that it is not practical to find the exact solution, for that reason we have to resort to approximate or heuristic solutions of the clustering problem.

2.3 Other criteria

Many clustering criteria were summarized in [34], of which most of them were found to be NP-complete. Some clustering cost functions are briefly mentioned in the following.

Complete linkage agglomerative clustering algorithm uses as its cost function the maximum within cluster dissimilarity, which is called the *minimum diameter* clustering [65]. The objective can be formalized by finding the partitioning $P_k^* = \{S_1, S_2, \ldots, S_k\}$ from the collection of all k-partionings $\mathcal{P}_k$ such that [65]:

$$\min_{P_k \in \mathcal{P}_k} \max_{q=1,\ldots,k} \max_{\boldsymbol{x}_i, \boldsymbol{x}_j \in S_q} d(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{2.6}$$

In other words, the goal is to find such a clustering where the maximum inter-cluster distance over all clusters is minimized. It has been shown that for $k \geq 3$, the problem is NP-complete by reduction from the graph coloring problem [58, 65].

A simple algorithm to cluster graph vertices is to calculate the minimum spanning tree of the graph and cut $k-1$ longest edges from it [5, 62, 186]. MST clustering by cutting turns out to be the same as *single linkage* agglomerative clustering algorithm [62]. Edges in the MST that will be retained after clustering are called *consistent edges*, and the edges that will be cut out are called *inconsistent edges*. If the number of clusters is unknown, then one needs to find out the position in the sorted edge list where inconsistent edges end and the consistent edges start. Modeling consistent edge lengths by a truncated normal distribution, and then automatically obtaining the cut threshold was proposed in [77].

In the k-median problem [78], the goal is to find k input vectors that minimize the following expression:

$$\sum_{\boldsymbol{x} \in X} \min_{\boldsymbol{c} \in C} \|\boldsymbol{x} - \boldsymbol{c}\|^2, \tag{2.7}$$

where $C \subset X$. The problem is also known as the *discrete median* problem, in contrast to the spatial median case where prototypes are not restricted to be from the input data set. The problem is NP-complete in Euclidean space [139]. In practice, this version of clustering cost is used when the optimal prototype is either difficult or meaningless to compute. A practical heuristic in that case is the *partition around medoids* (PAM) method [93], which is also known as the k-medoids algorithm. It is otherwise exactly the same as k-means, but the step for determining the optimal centroid has been replaced by a step where such an input element in the cluster is selected as the prototype that it minimizes the cluster distortion (2.7).

A clustering cost function that is specifically designed for hierarchical clustering is proposed in [45]. The goal is to generate such a graph partitioning where the length of the minimum spanning tree inside each cluster is minimized. It is shown in [45] that such a clustering problem is NP-complete. An approximation algorithm is also proposed that working in $O(N \log N)$ time and is within a factor of 3.42 from optimal clustering.

Previously, we have considered a case where the weights of the graph consist of real numbers, as defined by the l_p distances. However, one can also cluster

unweighted graphs, for example when clustering a social network data. In that case, distances between the observations are binary, a person $\boldsymbol{x}$ either knows a person $\boldsymbol{y}$ or does not. The squared error criterion is not sound in this case. Possible clustering criteria for this kind of data are *modularity cost* [15] and a objective function based on *dominant sets* [141]. The graph clustering problem is NP-complete for both of these objectives.

2.4 k-means algorithm

Perhaps, the most well-known clustering algorithm is the *k-means* [123], mainly because of its simplicity. It is also known as *generalized Lloyd algorithm* (GLA) [115], *hard c-means* and *Linde, Buzo, Gray* (LBG) algorithm. It has been applied in numerous different fields, such as statistics [165]. It is a practical solution to the optimization problem posed in (2.3).

The operation of the k-means algorithm is based on the fact that a globally optimal solution has to fulfill two *necessary* conditions [60]:

- *Nearest neighbour condition*: given cluster centroids C, optimal assignment of a data vector to the cluster is to its closest cluster prototype:

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}\in C} \|\boldsymbol{x} - \boldsymbol{c}_j\|. \tag{2.8}$$

- *Centroid condition*: given a partition $S_i = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{|S_i|}\}$, the optimal prototype is the partition centroid:

$$\boldsymbol{c}_i = \frac{1}{|S_i|} \sum_{j=1}^{|S_i|} \boldsymbol{x}_j. \tag{2.9}$$

There are several different strategies to prove the centroid condition (see [60] for three proofs). An interesting proof, which does not employ a direct use of derivatives, is obtained by considering the following equality [161]:

$$\sum_{i=1}^{|S_j|} \|\boldsymbol{x}_i - \boldsymbol{z}\|^2 = \sum_{i=1}^{|S_j|} \|\boldsymbol{x}_i - \boldsymbol{c}_j\|^2 + |S_j|\|\boldsymbol{c}_j - \boldsymbol{z}\|^2, \tag{2.10}$$

where $\boldsymbol{z}$ is any vector in $\mathbb{R}^D$. Since, the last term is non-negative, the minimum distortion is obtained when $\boldsymbol{c}_j = \boldsymbol{z}$.

The third necessary optimality condition is called *zero probability condition* [60]. It states that observations at the Voronoi face, occur at zero probability. If it

happened, there would be more than one cluster where the input vector could belong to, and the order of processing the data set will affect the solution. In practice, with real valued feature vectors the zero probability condition is true.

The k-means algorithm is a *local search* heuristic where the current solution is iteratively improved by alternating between nearest neighbour condition and centroid condition until convergence. When converged, centroids are generators of the centroidal Voronoi diagram [39]. The algorithm starts from an initial solution C, which can be randomly drawn vectors from X, or an output of any clustering algorithm minimizing the squared error. The effectiveness of different initialization strategies has been studied empirically for example in [142]. In case of more advanced clustering techniques, everything can be seen as an initialization to k-means, which as a local search heuristic can never give a worse solution. In any iteration, a lower bound on the locally optimal solution can be calculated in $O(N \log N + ND)$ time [191].

Convergence in finite number of steps follows from the fact that in each iteration the cost strictly reduces, so each solution can be seen only once during the execution of the algorithm. Since each solution induces a Voronoi diagram, for which we already have upper bound $O(N^{kD})$, k-means will converge in at least this many iterations. However, convergence to a globally optimal solution is not guaranteed. Actually, examples can be generated where the performance of the k-means is arbitrary bad, depending on the initialization [4, 67]. In practice, it is better to run k-means multiple times in parallel with different random initialization, and pick the solution with the least cost. This approach is known as the *multi-start k-means* [70]. The algorithm can be made significantly faster by using the lower bound on the locally optimal solution and stopping early if the current random start will not provably lead to a better solution than the best of previously computed solutions [191].

The time complexity of the standard version of k-means is $O(NkI)$, where I is the number of iterations needed for the convergence. It is an interesting task to try to analyze the worst case and the average case convergence speed of the algorithm. Har-Peled and Sadri [67] analyzed the one-dimensional case and found out that the upper bound on the number of iterations $O(N\Delta^2)$, does not depend on the number of clusters, but it depends on the spread of the data set Δ. In one-dimensional case, if the distance between the closest pair is normalized to unity, Δ is the diameter of the input data set. Recently, Arthur and Vassilvistkii [3] constructed an example, where k-means takes $2^{\Omega(\sqrt{N})}$ iterations.

In [14], it was proven that the convergence behaviour of the k-means is the same as Newton optimization algorithm. For a quadratic cost function, Newton algorithm finds the optimal solution in one step. Unfortunately, in finding a non-quadratic case locally optimal solution demands a super-linear time, and this analysis applies to k-means [14]. It has been noted that the convergence speed is affected by the initializa-

tion strategy, if initialization is performed more intelligently than random sampling from uniform distribution, then less iterations are needed for convergence [4].

Standard implementation of the k-means uses $O(kN)$ time per iteration. The nearest centroid search can be performed faster by placing the centroids to a spatial indexing structure [61, 90, 120, 134, 144]. Although, spatial indexing structures such as kD-tree [57] do scale well to a large number of cluster centroids, they do not scale to a high-dimensional case when $D > 8$ [144]. Then nearest centroid search will approach linear time instead of logarithmic time as promised by the search structure. The number of distance computations can also be reduced by considering that movement of a centroid is a *local* operation and it affects the neighbouring clusters only. A method utilizing this property is known as *code vector activity detection* [95]. Using the triangle inequality, bounds for both the partition and the centroid step can be computed resulting additional saving in the number distance calculations [43].

2.5 Exact and approximation algorithms

In addition to the exact clustering algorithm proposed by Inaba *et al.* [82], a number of other exact algorithms have also been proposed. One of the first exact algorithms for the squared error clustering problem was a dynamic programming algorithm by Jensen [85]. A branch-and-bound algorithm was proposed in [55, 107]. An optimization theoretic approach for (2.3) was formulated as an integer programming problem [150] and a constrained hyperbolic 0/1-program [132]. An interior point method was used in [132] to solve the problem with the help of branch-and-bound and numerous other mathematical programming tools. Interestingly, the authors managed to obtain optimal clustering to fairly large data sets, e.g. Fisher's iris data set consisting of 150 vectors. A method with much simpler implementation was designed by Brusco [18]. His method uses the repetitive branch-and-bound to achieve similar efficiency as in [132]. In [145], it was proven that the clustering problem can be formulated as a concave minimization problem, where every local minima is an integer solution. Relaxed version of the above mentioned algorithm yields a non-optimal but practical algorithm [145].

However, some recent theoretical work in approximation algorithms has shown that, at least theoretically, a *polynomial time approximation scheme* (PTAS is possible for the clustering problem [1, 49, 129, 137]. A PTAS is an algorithm, which takes as its input a data set and a user selected parameter ϵ. Parameter ϵ gives the desired maximal error of the approximation. A PTAS has polynomial time complexity, with all values of ϵ [176]. Unfortunately, none of the PTAS clustering algorithms have turned out to be practical because of large hidden constants [67] and, therefore,

practitioners use heuristic algorithms without performance guarantees. There is a big gap between theory and practice in the field of clustering algorithms [137].

On the other hand, practical clustering algorithms with proven worst case approximation ratio, have been published. *Randomized local search* (RLS) [53] is a clustering algorithm where local optima are avoided by perturbing the solution by swapping one centroid to a new location. A deterministic variant of the swapping algorithm was analyzed in [91] and it turns out to be $(25 + \epsilon)$-approximation algorithm. However, authors also propose a multi-swap variant, which achieves $(9 + \epsilon)$-approximation in the worst case [91]. Improving the randomized initialization of the k-means algorithm yields in terms of expectation an $8(\ln k + 2)$-approximation algorithm [4].

2.6 Agglomerative algorithms

An agglomerative clustering [13, 88] algorithm constructs the clusters by a sequence of merge operations. The agglomeration process starts by initializing each data vector as its own cluster. Two clusters are merged at each step and the process is repeated until the desired number of clusters has been obtained. *Ward's method* [182], also known as *pairwise nearest neighbour* (PNN) [47], selects the cluster pair to be merged in such a way that the square error criterion is least increased. In total, $O(N)$ iterations are needed, and in each iteration just enumerating all pairwise distances take $O(N^2)$, and so yields $O(N^3)$ time complexity in total. When considering the operations in the iterations, it turns out that most computation originates from distance calculations [158]. For arbitrary dimension D, this leads to $O(DN^3)$ time complexity. We call this method *exact PNN* [47].

Several speed-ups have been proposed in the literature to the exact PNN. One possibility to reduce the distance calculations is to use a matrix of pairwise distances. The matrix is upper triangular as the merge costs are symmetric. When two partitions S_i and S_j are merged, only row i and column j must to be recalculated, which leads to $O(N)$ update operations. Authors in [177] propose to use exact PNN variant with distance matrix for color quantization. Kurita [111] proposed to use distance matrix for updates and a heap structure for finding the minimum pairwise distance in $O(\log N)$ time. The time complexity for Kurita's method is $O(N^2 \log N)$, but unfortunately, keeping all pairwise distances in the heap and in the distance matrix the causes space complexity to $O(N^2)$.

Fast exact PNN [52] has linear space complexity, but is still an order of magnitude faster than the previous versions of PNN. Here, the idea is to maintain the nearest

neighbour pointer nn() to all partitions:

$$\mathrm{nn}(S_a) = \arg \min_{i \in [i,N], i \neq a} d_{\mathrm{PNN}}(S_a, S_i) \tag{2.11}$$

and use it to find the cluster pair to be merged fast. To find the minimum distance only $O(N)$ steps are needed in each iteration. After a merge operation, the nearest neighbour pointers must be updated to point to a new possibly closer cluster. This method is faster than the previous ones but its time complexity is still $\Omega(\tau N^2)$, where τ denotes the number of partitions whose nearest neighbour pointer must be updated. A similar idea was presented in [135, 42]. In their approach, a cluster pair to merge is searched from on the list of nearest neighbour pointers, and such clusters pairs (S_a, S_b) are merged which are $\mathrm{nn}(S_a) = b$ and $\mathrm{nn}(S_b) = a$. Unfortunately, the authors did not provide any time complexity analysis of their method.

Another possibility to speed up the operation of PNN is to use a dynamic closest pairs data structure, where update cost is $O(N \log^2 N)$ [44]. The total time complexity for clustering is then $O(N^2 \log^2 N)$ and the space complexity is $O(N)$. A Speed-up of around 35% can also be gained compared to the fast exact PNN by deferring the distance computations until necessary [94, 25]. Practical methods to achieve 10 to 15% reductions in running time have been proposed in [180]. All these methods still require quadratic time.

The $O(N^2)$ time complexity barrier can be broken by introducing an approximation to the search for the minimum merge cost [**P1**]. The idea is to restrict the search for the closest cluster pair on the k-nearest neighbours of each cluster. The time complexity of the algorithm is then improved from $O(\tau N^2)$ to $O(\tau N \log N)$ at the cost of a slight increase in distortion.

2.7 Graph clustering

There also exists a number of graph clustering algorithms [157] that are based on graph theoretic properties instead of optimizing some defined optimization criterion. Properties that have been used are for example, *connectivity* [17], *Szemerédi Regularity* [162] and *scale-free minimum spanning tree* [138].

Graph clustering based on connectivity starts with taking the complete graph and restricting it based on some rule. A typical restriction is the k-nearest neighbour graph. The clusters are then defined to be the connected components of the graph [17]. *Mutual k-nearest neighbour* graph (MkNNG) based clustering [17] defines a new graph based on a directed k-nearest neighbour graph where directed edges are turned to undirected by removing all those edges that point to one direction, only. Then, connected components of the graph are clusters and isolated vertices are outlier points. It has been shown that if clusters are separated, setting $k = \log(N)$

will give a correct clustering [17]. MkNNG clustering algorithm was generalized to a mutual range graph in [27]. *Range graph* is a proximity graph where all data objects that are closer than a user specified threshold from the object in question are considered neighbours. In this variant of MkNNG, a directed range graph is calculated first and it is then converted into a mutual one.

In CHAMELEON [92], a k-nearest neighbour graph is formed from the similarity matrix. Clustering is then performed in two stages: the k-nearest neighbour graph is divided into a relatively large number of sub-clusters and small sub-clusters are then iteratively processed with agglomerative clustering to obtain the final clustering. Graph theoretic clustering algorithm by Jarvis and Patrick [84] on the other hand defines, instead of kNNG, a *shared neighbourhood* graph. Shared neighbourhood between two elements stands for the number of same elements contained in their k-nearest neighbour lists. As in CHAMELEON, clustering is then performed on the shared neighbourhood graph. In a sense, the shared neighbourhood graph is a weighted version of MkNNG.

2.8 Gaussian mixture modeling

Another way to think of clustering is to assume that the data set has been sampled from a parametric generative distribution. This is called *model clustering* or in the statistics literature *density estimation*. The most common mixture model is *Gaussian mixture models* (GMM) [12], which is our concentration from here onwards. The most popular cost function is *maximum likelihood* (ML), where such a model is found that maximizes the likelihood of the observations. Gaussian mixtures can also be estimated by optimizing some information theoretic criteria, such as *minimum message length* (MML) [51]. By minimizing the message length, it is possible to simultaneously solve the number of components and the GMM parameters. In the following, we will only discuss the maximum likelihood case.

In this section, we assume that data has been sampled from k multivariate Gaussians, which together form a mixture model. Each Gaussian component is parametrized by its prior weight, mean vector and covariance matrix $\theta_i = (\pi_i, \boldsymbol{\mu}_i, \Sigma_i)$, and the full parametrization of the whole model is $\Theta = (\theta_i)_{i=1}^k$. Diagonal covariance matrices are mostly used in situations where there is a need to optimize the speed of the computation. The reason is that, computing the log-likelihood needs the inverse of the covariance matrix.

We also make the so called *incomplete data* assumption, where each data vector is assumed to be generated by one of the Gaussians but the class label is not available, hence the term incomplete data. The form of the Gaussian probability density

function is:

$$p(\boldsymbol{x}|\Theta) = \sum_{i=1}^{k} \pi_i p(\boldsymbol{x}|\theta_i), \tag{2.12}$$

where $\boldsymbol{x}$ is a vector in $\mathbb{R}^D$. Then, given N independent and identically distributed (i.i.d.) samples, the complete-data log-likelihood is expressed as:

$$\begin{aligned} \log p(X|\Theta) &= \log \prod_{j=1}^{N} p(\boldsymbol{x}_j|\Theta) \\ &= \sum_{j=1}^{N} \log \sum_{i=1}^{k} \pi_i p(\boldsymbol{x}_j|\theta_i), \end{aligned} \tag{2.13}$$

where $\sum_{i=1}^{k} \pi_i = 1$. Now, the maximum likelihood parameter estimate $\hat{\Theta}_{\mathrm{ML}}$ can be expressed as:

$$\hat{\Theta}_{\mathrm{ML}} = \arg\max_{\Theta} \log p(X|\Theta) \tag{2.14}$$

Unfortunately, no closed form solution exists to (2.14), and therefore an algorithmic solution has to be devised to solve the problem. A locally optimal iterative scheme is known as *expectation maximization* (EM) algorithm [37]. Even though global optimality of the ML cost function cannot guaranteed [12], something is known theoretically about the global optimality. If Gaussian components are well-separated, then the mean vectors can be tractably estimated by projection based methods [2, 31, 178] and even using EM [32].

EM can also be seen as a generalization of k-means algorithm [12]. It works by alternating between two steps, finding the posterior probabilities (*responsibilities*) for each observation and re-estimating the model parameters given previously obtained responsibilities [12]:

- **E step:** evaluates responsibilities using the current parameter values:

$$\gamma(z_{ij}) = \frac{\pi_j N(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \Sigma_j)}{\sum_{j=1}^{k} \pi_j N(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \Sigma_j)}. \tag{2.15}$$

- **M step:** Re-estimates the model parameters given the current responsibilities:

$$\boldsymbol{\mu}_j^{\mathrm{new}} = \frac{1}{N_j} \sum_{i=1}^{N} \gamma(z_{ij}) \boldsymbol{x}_i, \tag{2.16}$$

$$\Sigma_j^{\mathrm{new}} = \frac{1}{N_j} \sum_{i=1}^{N} \gamma(z_{ij}) (\boldsymbol{x}_i - \boldsymbol{\mu}_j^{\mathrm{new}})(\boldsymbol{x}_i - \boldsymbol{\mu}_j^{\mathrm{new}})^T, \tag{2.17}$$

$$\pi_j^{\text{new}} = \frac{N_j}{N}, \tag{2.18}$$

where

$$N_j = \sum_{i=1}^{N} \gamma(z_{ij}). \tag{2.19}$$

In the above, $\gamma(z_{ij})$ is the responsibility of vector $\boldsymbol{x}_i$ belonging to the Gaussian component j. It can be seen as the generalization of the discrete partition labeling of the hard clustering. In the hard clustering case, $\gamma(z_{ij}) = \{1, 0\}$; it is one when the observation belongs to the cluster, and zero when it does not.

The EM algorithm provides a locally optimal solution to the log-likelihood maximization problem. The quality of the solution generated by EM depends on the initial solution [130]. To overcome this problem, a number of heuristics have been proposed. The most common way to initialize the EM algorithm is to first run hard clustering algorithm (e.g. k-means) on the data set and then to interprete each cluster as a Gaussian component. Component parameters are calculated from relative numbers of observations (mixing weight) and centroid (mean vector), and the covariance matrix is calculated from the observations in the cluster.

$\log p(X|\Theta) = -26.8$ $\log p(X|\Theta) = -26.6$

Figure 2.3: Example of GMM with diagonal covariance matrices: random initialization tuned by 10 k-means iterations (left) and a locally optimal output of the EM algorithm (right).

The *Split-and-merge EM* (SMEM) algorithm [173] performs the EM algorithm after local convergence and selects three components for a split-and-merge procedure. One of these component is split in two halves, and the other two components

$\log p(X|\Theta) = -26.6$ $\log p(X|\Theta) = -26.5$

Figure 2.4: Example of GMM with full covariance matrices: random initialization tuned by 10 k-means iterations (left) and a locally optimal output of the EM algorithm (right).

are merged. Minagawa *et al.* [133] noted that the acceptance rule of the SMEM algorithm is incorrect, resulting to an accidentally discarded globally optimal solution. Merge is defined in a theoretically sound manner in the SMEM algorithm, whereas split is a heuristic operation. For this reason, a merge based algorithm was proposed by Verbeek *et al.* [179]. Their algorithm iteratively inserted components into the mixture until a desired number of Gaussian components is obtained. After each merge, EM is run again until convergence.

Genetic algorithms (GA) have also been used to estimate the GMM parameters [128] when maximum likelihood is used to measure the fitness of a solution. A genetic algorithm using *Minimum description length* (MDL) based fitness function has also been proposed in [146] where the number of components is automatically set. In these approaches, GMM parameters were encoded into the chromosome, but a different encoding is proposed in [166]. A chromosome codes the indices of the generating mixture components for the vectors. In a sense, the proposed approach tries to estimate the Gaussian mixture parameters directly by encoding the latent variables in the individuals of the GA.

It has been noted that sometimes EM converges slowly to a locally optimal solution [50, 26]. Especially, EM will slow down considerably if it hits almost the plateau part of the fitness landscape [163]. Alternative strategies that maximize the parameters sequentially, instead of simultaneously, have been proposed as *Space-alternating generalized EM* (SAGE) [50] and *Component-wise EM* [26].

2.9 Outlier detection

Outlier detection refers to methodologies for eliminating observations from the input data set that adversely affect the statistical modeling [76]. *Inliers*, on the other hand, are the observations one is looking for, and those should be retained for the modeling. Numerous clustering methods include outlier detection as an additional step in the modeling process [17, 41, 63, 64, 76, 86, 189]. In general, an input data object is designated as an outlier if it does not *fit* to the model being constructed [76].

Another point of view to outlier detection is to see them not as noise to be eliminated but as interesting and surprising observations in need of more study. This approach is taken in data mining and knowledge discovery. A mixed case can also happen, where inlier observations are corrupted by noisy observations, and the task is still to find surprising observations [119]. Some applications of outlier detection in knowledge discovery include intrusion detection in computer network security [112, 114, 140] and abnormal human-activity detection [188].

Outlier detection methods can be divided into two categories: *supervised* or *unsupervised*. Supervised methods require external training data, where data vectors are labeled as inliers or outliers. Supervised outlier detection is also known as *novelty detection* [127]. Examples of the two-class classification approach applied to outlier detection are for example *neural networks* [118, 184], neural networks with *radial basis function* (RBF) with *principal component analysis* (PCA) *dimensionality reduction kernel* [117] and *k-NN classifier* [114].

The two-class classification approach has problems; it is implicitly defined that all relevant types of outliers have already presented in the training data set. One-class training is the other possibility where only examples from the inlier class are used in the training. In the classification step, an unknown observation is scored against the inlier class and the observation is decided to be an outlier if its score is below a user specified threshold. These methods are for example one-class *support vector machines* (SVMs) [122, 167] or one-class SVM with kernel *maximum likelihood linear regression* (MLLR) adaptation [188]. A hybrid method is one where the classifier does not rely on outlier examples on training step but can improve the classification accuracy if examples are available [23].

In unsupervised outlier detection, we have a data set without any separate training set with labeled examples. In practice, unsupervised methods need an exact definition of what is an outlier. The differences between the methods are then mostly related to the differences in the definitions. Intuitively, an outlier should be an observation that is far from other observations, as an example see Fig. 2.5. The figure represents astronomical observations of different stars. Ground truth labeling of the observations to inliers and outliers was done by a trained astronomer. Inlier observations represent stars in the so called main sequence and outliers are stars,

Figure 2.5: Example of a data set with normal observations and an outlier observations. Shown also outliers detected by [**P2**] with two different neighbourhood sizes.

which do not fit to that pattern.

Definitions of unsupervised outliers fall roughly into five categories [87]: i) *distribution-based*, ii) *clustering-based*, iii) *depth-based*, iv) *distance-based* and v) *density-based*. Distribution-based methods originate from statistics where an observation is considered as an outlier if it deviates too much from the underlying distribution. For example, in normal distribution an outlier is an observation whose distance from the average observation is three times of the variance [56]. The problem is that in real world cases the underlying distribution is usually unknown and cannot be estimated from the data without outliers affecting the estimate. Clustering-based methods work similarly: the whole data set is clustered and observations that do not fit to the overall clustering pattern are decided as outliers [86, 187, 190][**P3**]. As in distribution-based methods, outliers affect the clustering model. Depth-based outlier detection methods compute different layers of D-dimensional convex hulls of the data set. Outliers are then defined to be in the outer layers of these hulls [87].

Distance-based methods [105, 106] define an outlier as an observation that is at least at distance $d_{\min}$ away from p percentage of observations in the data set. The problem is then finding an appropriate settings of $d_{\min}$ and p such that outliers would be correctly detected with a small number of false detections. This process usually needs domain knowledge [106]. Typical distance-based methods require

$O(N^2)$ distance computations, but a faster implementation has been reported in [9].

In density-based methods, outliers are detected from local density of observations. These methods use different density estimation strategies. A low local density of the observation is an indication of a possible outlier. Ramaswamy *et al.* [147] proposed a method, in which n largest kNN distances are considered as outliers. This can be seen as *sparseness estimate* of a vector, in which the n sparsest vectors are considered as outliers. Other methods based on k-neighbourhood are presented in [16, 9, 87, 108, 190]. Graph theoretic methods infer the local density by using a k-nearest neighbour graph [17][**P2**]. As with distance-based methods, computation of the local density for all observations can be slow. Speed-up can be obtained by pruning the nearest neighbour search [87].

Chapter 3

Speaker Recognition

IN *speaker verification* [11], an unknown speech utterance is introduced to the system accompanied by a claim. The task is to decide whether the claim was true or false, by matching the unknown test utterance to a previously stored model. In *speaker identification*, on the other hand, the unknown speech utterance is matched against a database of known speakers. Output of the speaker identification system is the identity of the speaker in question, or in the case of *open-set identification*, the decision that the speaker is unknown to the system. Speaker recognition tasks can be further classified into *text-dependent* and *text-independent* tasks.

In text-dependent systems, the spoken text or password is fixed. In text-independent recognition, the user is free to speak anything as the enrollment utterance, and also in the using of the system. Applications of the text-dependent recognition are restricted to security and access control whereas text-independent recognition can be used in continuous recognition as well.

A speaker recognition system consists of two main components: i) *front-end* processing component and ii) pattern recognition component. Front-end processing reads audio data, preprocesses it, applies voice activity detection and feature extraction. The pattern recognition subsystem, handles features modeling and pattern matching. Feature extraction produces a sequence of feature vectors $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^D$. Each feature vector describes short-term signal statistics. The feature cloud from the utterance contains speaker discriminative information. The front-end system and pattern recognition system can be separately tuned.

3.1 Front-end processing

3.1.1 Voice activity detection

In the general terms, *voice activity detector* (VAD) [10] is a binary classifier that labels segments of the signal either as speech or non-speech. It is usually needed as the first component in voice-based applications such as automatic speech recognition, speech enhancement [148], forensic speech skimming [172] and speaker recognition [152]. A common property of these applications is that only human sounds (typically only speech) are of interest to the system, and it should therefore be separated from the background. In different applications, the granularity of the voice activity decision varies. We call *frame level VAD* a system which produces per frame decisions, and *end-point detector* a system which detects continuous speech segments.

Voice activity detection is an essential part of any speaker recognition system [169]. It is clear that including non-speech frames in the modeling process would bias the resulting model, especially if the number of non-speech frames is significant. It is also known that not all speech frames have equal discriminative power as voiced phonemes are more discriminative than unvoiced phonemes [156]. It has been found that depending on the speech corpus, a badly optimized VAD can lead to a catastrophic detection accuracy: from 17% error rate to nearly the accuracy of just coin flipping [**P7**]. A similar result has been recently seen in the *speaker diarization systems* [171] where the biggest source of diarization error is contributed by errors made by the VAD preprocessing component [80].

Numerous approaches for voice activity detection exist in the literature and exhaustive categorization of all those approaches is not easy. However, voice activity detectors fall into two categories: i) those that threshold short-term signal statistics, and ii) those that use machine learning approaches. The VADs in the first group perform thresholding according to *spectral likelihood ratio* [89], *periodicity* [**P7**], *energy* [170] and *long-term spectral divergence* (LTSD) [148] just to mention a few.

3.1.2 Feature Extraction

In speaker recognition, features that capture the anatomical and behavioral characteristics of the speaker are desired. Speech itself is highly complex signal, which carries several features mixed together [154]. Because of the high complexity, it is difficult to realize the features that robustly capture the speaker individuality [97].

Features can be divided into three categories: *segmental*, *suprasegmental* and *high-level* [151]. Segmental features contain speech signal characteristics computed over short segments of 10-30 milliseconds in duration. Suprasegmental features, on

the other hand, contain information that spans multiple segments, sometimes the whole utterance. High-level features are discrete in nature, for example linguistic features such as idiosyncratic word usage or phonetic features such as pronunciation style. In the following, we concentrate on the segmental and suprasegmental features.

3.1.3 Segmental features

In segmental feature extraction, the speech signal is analyzed in short segments or *frames* where local stationarity is assumed. Frame length is typically 10-30 milliseconds, and frames overlap 25-50% of the frame length [98]. A single feature vector is computed from each frame individually. The *mel-frequency cepstral coefficients* (MFCCs) [33] are the most popular spectral features used in speaker recognition systems [19].

The steps in the MFCC processing are summarized in Fig. 3.1. Time-windowing is performed first to suppress discontinuities at the frame boundaries, followed by the magnitude spectrum computation using *discrete Fourier transform* (DFT) [136]. Magnitude spectrum is obtained from the complex valued Fourier coefficients by taking the absolute value of it. Filterbank processing provides dimensionality reduction by creating a smoother version of the original magnitude spectrum. Non-linear frequency warping is obtained by placing the center frequencies of each filter according to the defined warping function. It is designed so that it will emphasize lower frequency part of the spectrum. Warping function used in MFCC is the so called *mel* or *melody* scale [69]. The cepstrum is finally obtained by first applying logarithm and then the *discrete cosine transform* (DCT). The zeroth coefficient corresponds to the frame energy, and it is dropped. Typically 10-20 low-order coefficients are retained for further processing.

Figure 3.1: Work flow of the MFCC signal processing.

3.1.4 Suprasegmental features

Suprasegmental features are typically calculated over the whole utterance. *Long-term average spectrum* (LTAS) [79] is defined as an average magnitude spectrum of the whole utterance. An estimate of the average magnitude spectrum can be computed by first dividing the speech signal into overlapping frames and then computing the magnitude spectrum for each frame separately, followed by time averaging [183].

LTAS is a n-dimensional feature vector where n is the FFT binsize. It has been mostly used in forensic speaker recognition [155] as it is easy to compare two different LTAS vectors [116] visually. Example on how different LTAS vectors can be visually evaluated is shown in Fig. 3.2. It shows four LTAS vectors calculated from four different subjects over telephone line. It has also been used in early automatic speaker recognition studies [126, 68]. However, LTAS is of lower accuracy than MFCC, and does not provide substantial improvement when used in combination with (classifier fusion) MFCC [**P6**]. In terms of speaker modeling and matching, LTAS is straightforward to compute. LTAS vectors can be directly matched using any similarity or dissimilarity measure. Similarity functions that have been used in LTAS based speaker recognition include Euclidean distance, cosine similarity, correlation and Kullback-Leibler divergence [**P6**] and also a recent study [175] used *standard deviation of difference distribution* (SDDD) [68] index.

Figure 3.2: Example of LTAS calculated from four different speakers.

Long-term features calculated from the so called *prosodic* parameters have also been used in speaker recognition. Prosody refers to the rhythm, stress and intonation of speech. Most common prosodic features describe syllable length, loudness and pitch (or F0). F0 and intensity contours have been used in text-dependent speaker recognition [6]. In the text-independent case, long-term F0 distributions contain speaker discriminative power [99]. Distributions can be matched, either by their parameter vectors or by histogram matching [99]. Histogram matching with Kullback-Leibler divergence was found to be more accurate. However, a histogram is a discrete model of the long-term distribution and a continuous model such as kernel density estimator could provide better accuracy [104]. In [36], prosody features were extracted by first segmenting the whole utterance into pseudo-syllables using energy contour. From each pseudo syllable, a sixth-order Legendre polynomial was fitted

to the log F0 and log energy contours. Legendre polynomial coefficients, in addition to segment duration, were then modeled by GMM using a session variability compensation [96].

3.2 Speaker modeling

Speaker recognition systems have typically used generative models such as vector quantization [11, 160] (aka the *centroid model*) and Gaussian mixture models [153], or discriminative models such as *support vector machines* (SVMs) [21] and *neural networks* [48]. Approaches that combine the generative and discriminative models have been recently introduced [22]. When using generative models we assume that there is an unknown distribution where the feature cloud is sampled from. The goal in the modeling is then to estimate the parameters as well as possible. In discriminative training, the goal is to find *decision boundary* that classifies unseen samples as well as possible. The basic difference is that discriminative modeling needs examples from the negative (or *impostor*) class, whereas generative approach works only with the speakers own feature vectors. In the following, we restrict the discussion to the generative models.

Figure 3.3: Speaker model training in the maximum likelihood approach.

A generative model is typically trained using the *maximum likelihood* (ML) principle where the speaker model is optimized by only analyzing the training data provided by the user. In the case of GMM training, statistical independence of the feature vectors is assumed, and the log-likelihood (2.14) is therefore used. In VQ, mean squared error (2.4) is the cost function to be optimized. System diagram of the maximum likelihood approach is shown in Fig. 3.3. The ML approach usually does not generalize well to unseen speech data with a finite amount of training material [12].

The size of the centroid set C (number of Gaussian components, in the case of GMM) defines the *model order*, and it is a user-selectable parameter. Increasing the model order creates more accurate estimation of the underlying distribution and decreases recognition errors with increased processing time. Too large models will over-fit the training data and degrade the speaker recognition accuracy [102, 103].

Maximum a posteriori (MAP) training [59] attacks the problem of limited training data by restricting how closely the model is allowed to be optimized for the

Figure 3.4: Speaker model training by the MAP principle.

training data. In the MAP approach, prior knowledge of the distribution of the model parameters is incorporated into the modeling process. Even if some areas of the feature space are less represented in the training data, the prior information about the parameters can help to overcome the problem. However, incorporating the prior information is not trivial because prior parameter distribution has its own parameters, known as *hyperparameters*, which can be difficult to estimate. In the GMM case, *universal background model* (UBM) [152] was proposed as a practical way to set the hyperparameters. In VQ, similar approach with UBM as a hyperparameters to the centroid set is proposed in [**P4**]. A system diagram of the maximum *a posteriori* approach using UBM-based prior distribution parameters is shown in Fig. 3.4.

The UBM for GMMs is calculated by optimizing the maximum likelihood principle using EM algorithm. In VQ, the UBM is calculated by optimizing the MSE cost function. Individual speaker models are then adapted from the UBM using a modified MSE criterion by applying k-means or EM algorithm.

3.3 Mismatch compensation

Errors in speaker recognition accuracy are mostly due to mismatch between testing and training conditions. With TIMIT corpus, which can be obtained from *Linguistic Data Consortium* (LDC)[1] , which contains only noise free and single microphone recordings, a MFCC based system is able to achieve 0% error rate [97]. On the other hand, when channel mismatch is present, the error rate can increase considerably. For a practical speaker recognition system it is necessary to include some form of mismatch compensation technology.

[1]http://www.ldc.upenn.edu/

Various approaches have been proposed for tackling the channel mismatch problem, including robust feature extraction [125], feature normalization [143], model transformation [96, 181, 168], and match score normalization [7, 152]. Feature and speaker model transformations, including *affine transformation*, have been studied by different authors [124, 125, 181, 159]. Most common mismatch compensation techniques are *eigenchannel* [96] for Gaussian mixture models and *nuisance attribute projection* (NAP) [19] for SVMs. In both cases the model parameters are tuned using the parameters of the channel and session variability transformation.

Above mentioned channel compensation methods usually require either parallel training data recorded simultaneously through various handsets, or a large number of training utterances collected from multiple recording sessions from a number of speakers. These data sets are then used for estimating the transformation parameters. Instead of learning the transformation parameters from an external corpus, another approach is to design a matching scheme that is invariant to certain transformations in the feature space. For example, the method proposed in [**P5**] is designed to be invariant to rotation, translation and uniform scaling.

Chapter 4

Summary of the Publications

IN **the first paper** [**P1**], we proposed a fast agglomerative clustering method using a k-nearest neighbor graph. The graph is used to restrict the search of the cluster pair with the smallest agglomerative merge distance. Agglomerative clustering is a well known method for its ease of implementation and quality in terms of MSE. Unfortunately, the original method is slow, $O(N^3)$, which has later been lowered to $O(\tau N^2)$. In this work, we managed to break the $O(N^2)$ time complexity barrier by introducing an approximation to the search of the minimum merge cost. The time complexity of the algorithm is improved from $O(\tau N^2)$ to $O(\tau N \log N)$ at the cost of a slight increase in distortion. Here, τ denotes the number of nearest neighbor updates required at each iteration. According to our experiments, a relatively small neighborhood size is sufficient to maintain the quality close to that of the full search.

In the second paper [**P2**], we apply the k-nearest neighbour graph from [**P1**] to unsupervised outlier detection. We define an observation to be an outlier if, in a directed k-nearest neighbour graph, it is not a neighbour of other observations. We also propose a modification to the existing kNN distance -based method, so that user does not need to know in advance the number of outliers present in the data set. We compared the methods with real and synthetic data sets. The results show that the proposed method achieves reasonable results with synthetic data and outperforms comparative methods with real data sets when the input data set is small and performs comparatively for large datasets.

In the third paper [**P3**], we extend the k-means algorithm to robust mean vector estimation, and incorporate outlier removal into the process. In the case of overlapping symmetric mono modal distributions, k-means will include bias to the estimate of the mean. In this paper, we consider the overlapping areas of the distributions

as noise (or outliers), which is removed from the clustering process step by step. We present an outlier removal clustering (ORC) algorithm that achieves the above mentioned goal. The method employs both the clustering and outlier discovery operations to improve the estimation accuracy of the centroids of the generative distribution. The ORC algorithm consists of two stages. The first stage performs pure k-means algorithm, while the second stage iteratively removes vectors that are far from their cluster centroids. We ran a set of experiments on three synthetic data sets and three map images that were corrupted by lossy compression. The results indicate that the proposed method has a lower estimation bias on data sets with overlapping clusters than the compared methods.

In the fourth paper [**P4**], we formulate the speaker modeling task as a maximum *a posteriori* (MAP) adaptation of the VQ model. We rigorously formulate MAP adaptation for the VQ when the speaker model is adapted from UBM generated by any clustering algorithm. We show experimentally that VQ is comparable to GMM in terms of verification accuracy. Furthermore, the proposed speaker adaptation can be up to 20 times faster when using VQ than in the case of GMM.

In the fifth paper [**P5**], we design a speaker recognition system that tries to overcome technical mismatches between training and testing conditions. We propose a matching scheme, which is invariant to feature rotation, translation and uniform scaling. Channel compensation methods, such as eigenchannel [96] and NAP [19], need external training material to learn the channel factors. Whereas, we attempt to design a mismatch invariant system that operates without learning channel factors from the external training material. The proposed approach uses a neighborhood graph to represent the global shape of the feature distribution. The reference and test graphs are aligned by graph matching and the match score is computed using conventional template matching. Experiments on the NIST-1999 SRE corpus indicate that the method is comparable to the conventional GMM- and VQ-based approaches.

In the sixth paper [**P6**], we concentrate on the feature extraction part of the speaker recognition system. Most speaker recognition systems use the mel-frequency cepstral coefficients (MFCCs) to describe the spectral properties of speakers. In forensic phonetics, the long-term average spectrum (LTAS) has been used for the same purpose. It provides an intuitive graphical representation, which can be used for visualizing and quantifying differences between speakers. However, few studies have reported the use of LTAS in *automatic* speaker recognition. Thus, the purpose of the paper is to systematically study how to use the LTAS in automatic speaker recognition. We found out that it provides only marginal additional discriminative information in respect to the MFCC-based system. However, LTAS is very simple

to implement and it is order of magnitude faster in matching than MFCC-based system.

In the seventh paper [**P7**], importance of voice activity detection (VAD) for the performance of speaker recognition system is studied. We propose an improved voice activity detector based on periodicity information. A design requirement of the voice activity detector is that it needs to work in realtime environment, and to make speech/non-speech decisions without processing delay. Performance of the proposed method is compared against two existing methods: a realtime method based on long-term spectral divergence (LTSD) and a simple energy based method. The periodicity-based method outperforms the other realtime method (LTSD), and performs comparably with the energy-based method when applied for NIST 2001 and 2006 speaker recognition evaluation corpora. The method was also tested for speech-non-speech segmentation on surveillance, voice dialog and forensic recordings.

The contributions of the author of this thesis can be briefly summarized as follows. In [**P1**], the author designed, implemented and tested the first version of the method. In [**P2**, **P7**], author designed the algorithm, conducted most of the experiments and wrote the paper. In [**P3**, **P4**, **P5**], author was the main responsible of the algorithm development, implementation, writing the paper, and also participated in the experimentation. In [**P6**], author contributed in running the experiments and writing of the paper.

Chapter 5

Summary of the Results

IN this chapter, main results of the original publications [**P1**]-[**P7**] are summarized. Each paper included in this thesis contribute to some part of the general speaker recognition system. The proper place of each of the paper in the complete system is shown in Fig. 5.1. In this Figure, papers are denoted by [**P1**]-[**P7**]. Majority of the contributions are focused on the speaker modeling by cluster analysis [**P1**]-[**P4**], but contributions are also given in the voice activity detection [**P7**], feature extraction [**P6**] and pattern matching [**P5**] subsystems.

Figure 5.1: Original publications in the speaker recognition system.

Summarized results

P1: Over 120 times faster and the same clustering performance for BIRCH2 data set with DLA+k-means than with the PNN+k-means algorithm.

P2: Zero detection error with the ODIN method when applied to HR, NHL1 and NHL2 data sets and comparable performance with intrusion detection data set (KDD).

P3: About 1.5 - 2 times more accurate centroid estimation with the ORC method than with the baseline k-means.

P4: Twenty times faster speaker adaptation speed with NIST 2006 corpus using VQ-MAP than using GMM-MAP, while having exactly the same recognition accuracy (17% *equal error rate* (EER)).

P5: About 13% relative improvement in speaker identification performance when graph matching is fused with the GMM-MAP system.

P6: Over 300 times faster speaker identification with LTAS than with MFCC features. About 4% relative improvement in speaker verification accuracy when LTAS is fused with MFCC than for MFCC alone.

P7: About 63% relative improvement in speaker verification with NIST 2006 corpus if using VAD in comparison of not using VAD, and 53% relative improvement of using period VAD than using LTSD VAD.

Speech Corpora

In the speaker verification experiments, four different telephone corpora were used. All corpora were originally used as a part of the annual *National institute of standards and technology* (NIST) speaker recognition evaluations[1] . All data are conversational speech, with the sampling rate of 8 kHz and quantization 8-bit μ-law, which is a commonly used bit quantization technique in telecommunications. Corpora are summarized in Table 5.1. Difficulty of the evaluation has been increased after each new iteration of the competition.

[1] http://nist.gov/speech/tests/spk/

Table 5.1: Summary of the speech corpora

Description	Subset of NIST-1999 (I)	Subset of NIST-1999 (II)	NIST-2001	NIST-2006
Language	English	English	English	Multiple
Speakers	80	230	174	816
No. trials	950	52900	22418	53966
Telephone type	Land-line	Land-line	Mobile	Mobile
Handset mismatch	No	No	Yes	Yes
Non-speech removed	Yes	Yes	Some	No
Train speech	60 sec.	60 sec.	2 min.	5 min.
Test speech	60 sec.	60 sec.	30 sec.	5 min.
Publications where used	[**P5**]	[**P6**]	[**P4**, **P6**, **P7**]	[**P4**, **P7**]

Chapter 6

Conclusions

A PRACTICAL speaker recognition system consists of signal processing, modeling, matching and decision logic modules. In this thesis, we have studied all but decision logic, with special concentration on speaker modeling and matching based on vector quantization. One of our contributions is the formalization of a maximum *a posteriori* (MAP) vector quantization matching system using a universal background model. It achieves similar verification accuracy as Gaussian mixture model based systems but the adaptation is 20 times faster.

In our MAP adapted VQ system, the universal background model is estimated by a clustering algorithm that optimizes the model based on the squared error criterion. We gave contributions to UBM construction by designing a novel sub-quadratic time agglomerative clustering algorithm. We also contributed to the robust estimation of cluster parameters by outlier removal, first by using the k-nearest neighbour graph, and second by using a modified k-means algorithm.

Feature extraction issues were studied by experimenting on a long-term average spectrum feature with different matching techniques. We found out that LTAS is only marginally useful when combined with state-of-the-art GMM-MFCC system, but it is computationally efficient. We proposed a new realtime voice activity detection method for speaker recognition, and experimented with the effect of the voice activity parameters on the speaker recognition accuracy. Contrary to the earlier reports, we found out that the accuracy of the whole system is greatly affected by proper tuning of the VAD parameters. We noticed that the voice activity detector should be tuned so that as few non-speech frames are accepted as possible.

The performance of the speaker recognition system is affected by any kind of mismatch between the audio material used to train the speaker model and the test utterance. Mismatch can happen, for example, when transmission channel (microphone or audio coding system), language or emotion is different. In this thesis, we

contributed to the field of mismatch compensation by designing a restricted affine invariant modeling and matching scheme using the graph matching methodology.

As future work, one should study different ways to overcome exponential time complexity of the graph matching approach. By using a graph matching algorithm based on the *spectral graph theory* one could apply linear algebra techniques instead of combinatorial optimization methods. We would like to improve on the combinatorial matching algorithm where matching is now done by reduction to a clique problem and then solving the clique by a local search algorithm. Direct local search solution makes it possible to use larger graphs in matching. Another direction of future research would be to match SVMs using the MAP adapted vector quantization models. Eigenchannel channel compensation technique, originally proposed for the Gaussian mixture models, could be adapted to the vector quantization framework as well.

References

[1] Ackermann, M. R., Blömer, J., and Sohler, C. Clustering for metric and non-metric distance measures. In *ACM-SIAM Symposium on Discrete Algorithms (SODA 2008)* (San Francisco, California, January 2008), pp. 799–808.

[2] Arora, S., and Kannan, R. Learning mixtures of arbitrary Gaussians. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC 2001)* (2001), pp. 247–257.

[3] Arthur, D., and Vassilvistkii, S. How slow is the k-means method? In *Proceedings of the 22nd Annual Symposium on Computational Geometry (SCG 2006)* (June 2006), pp. 144–153.

[4] Arthur, D., and Vassilvitskii, S. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms (SODA 2007)* (New Orleans, Louisiana, 2007), pp. 1027–1035.

[5] Asano, T., Battacharya, B., Keil, M., and Yao, F. Clustering algorithms based on minimum and maximum spanning trees. In *Proceedings of the 4th Annual Symposium on Computational Geometry (SCG 1988)* (1988), pp. 252–257.

[6] Atal, B. Automatic speaker recognition based on pitch contours. *Journal of Acoustic Society of America 52*, 6 (1972), 1687–1697.

[7] Auckenthaler, R., Carey, M., and Lloyd-Thomas, H. Score normalization for text-independent speaker verification systems. *Digital Signal Processing 10* (2000), 42–54.

[8] Aurenhammer, F. Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Computing Surveys 23*, 3 (1991), 345–405.

[9] Bay, S. D., and Schwabacher, M. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD 2003)* (Washington, D.C., 2003), pp. 29–38.

[10] Benyassine, A., Schlomot, E., and Su, H. ITU-T recommendation g729 annex b: A silence compression scheme for use with g729 optimized for v.70

digital simultaneous voice and data applications. *IEEE Communications Magazine 35* (1997), 64–73.

[11] Bimbot, F., Bonastre, J.-F., Fredouille, C., Gravier, G., Meignier, S., Merlin, T., Ortega-Garcia, J., Magrin-Chagnolleau, I., Petrovska-Delacretaz, D., and Reynolds, D. A. A tutorial on text-independent speaker verification. *EURASIP Journal on Applied Signal Processing*, 4 (2004), 430–451.

[12] Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer-Verlag, New York, 2006.

[13] Boberg, J., and Salakoski, T. General formulation and evaluation of agglomerative clustering methods with metric and non-metric distances. *Pattern Recognition 26*, 9 (September 1993), 1395–1406.

[14] Bottou, L., and Bengio, Y. Convergence properties of the k-means algorithms. In *Advances in Neural Information Processing Systems (NIPS 1995)* (1995), pp. 585–592.

[15] Brandes, U., Delling, D., Gaertler, M., Görke, R., Hoefer, M., Nikolski, Z., and Wagner, D. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering 20*, 2 (February 2008), 172–188.

[16] Breunig, M. M., Kriegel, H. P., Ng, R. T., and Sanders, J. LOF: Identifying density based local outliers. In *Proceedings of the 19th International Conference on Management of Data (SIGMOD 2000)* (Dallas, Texas, 2000).

[17] Brito, M. R., Chávez, E. L., Quiroz, A. J., and Yukich, J. E. Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. *Statistics & Probability Letters 35*, 1 (August 1997), 33–42.

[18] Brusco, M. J. A repetitive branch-and-bound procedure for minimum within-cluster sums of squares partitioning. *Psychometrika 71*, 2 (June 2006), 347–363.

[19] Burget, L., Pavel Matejka, M., Schwarz, P., Glembek, O., and Cernocký, J. Analysis of feature extraction and channel compensation in a GMM speaker recognition system. *IEEE Transactions on Audio, Speech, and Language Processing 17*, 7 (September 2007), 1979–1986.

[20] Callahan, P. B., and Kosaraju, S. R. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the Association for Computing Machinery 42*, 1 (1995), 67–90.

[21] Campbell, W., Campbell, J., Reynolds, D., Singer, E., and Torres-Carrasquillo, P. Support vector machines for speaker and language recognition. *Computer Speech & Language 20*, 2–3 (April–July 2006), 210–229.

[22] Campbell, W. M., Sturim, D. E., and Reynolds, D. A. Support vector machines using GMM supervectors for speaker verification. *IEEE Signal Processing Letters 13*, 5 (2006), 308–311.

[23] Cao, L. J., Lee, H. P., and Chong, W. K. Modified support vector novelty detector using training data from outliers. *Pattern Recognition Letters 24* (2003), 247–2487.

[24] Capoyleas, V., Rote, G., and Woeginger, G. Geometric clustering. *Journal of Algorithms 12* (1991), 341–356.

[25] Cardinal, J., and Eppstein, D. Lazy algorithms for dynamic closest pair with arbitrary distance measures. In *Proceedings of the SIAM Workshop on Algorithm Engineering and Experiments (ALENEX04)* (New Orleans, January 2004), pp. 112–119.

[26] Celeux, G., Chretien, S., Forbes, F., and Mkhadri, A. A component-wise EM algorithm for mixtures. *Journal of Computational and Graphical Statistics 10*, 4 (December 2001), 697–712.

[27] Chávez, E. A subquadratic algorithm for clustering and outlier detection in massive metric data. In *Proceedings of the 8th International Symposium on String Processing and Information Retrieval (SPIRE 2001)* (Laguna de San Rafael, Chile, November 2001), pp. 46–58.

[28] Clarkson, K. L. Fast algorithms for the all-nearest-neighbor problem. In *Proceedings 24th IEEE Symposium Foundations of Computer Science (FOCS 1983)* (1983), pp. 226–232.

[29] Clarkson, K. L. Nearest-neighbor searching and metric space dimensions. In *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, G. Shakhnarovich, T. Darrell, and P. Indyk, Eds. MIT Press, 2006, pp. 15–59.

[30] Cormen, T. H., Leiserson, C. E., and Rivest, R. L. *Intoduction to Algorithms*. The MIT Press, 1998.

[31] Dasgupta, S. Learning mixtures of Gaussians. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS 1999)* (1999), pp. 634–644.

[32] Dasgupta, S., and Schulman, L. J. A two-round variant of EM for Gaussian mixtures. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI 2000)* (2000), pp. 152–159.

[33] Davis, S., and Mermelstein, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing 28*, 4 (1980), 357–366.

[34] Day, W. H. E. Complexity theory: An introduction for practitioners of classification. In *Clustering and Classification*, P. Arabie, L. Hubert, and G. De Soete, Eds. World Scientific Publishing Co., Singapore, 1992, pp. 199–233.

[35] De Loera, J. A., Hemmecke, R., Onn, S., Rothblum, U. G., and Weismantel, R. Convex integer maximization via graver bases. *Journal of Pure and Applied Algebra* (2008). to appear.

[36] Dehak, N., Dumouchel, P., and Kenny, P. Modeling prosodic features

with joint factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing 15*, 7 (September 2007), 2095–2103.

[37] Dempster, A., Laird, N., and Rubin, D. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B 39* (1977), 1–38.

[38] Drineas, P., Frieze, A., Kannan, R., Vempala, S., and Vinay, V. Clustering large graphs via the singular value decomposition. *Machine Learning 56*, 1–3 (July 2004), 9–33.

[39] Du, Q., Faber, V., and Gunzburger, M. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Review 41*, 4 (December 1999), 637–676.

[40] Dunford, N., and Schwartz, J. T. *Linear operators*, vol. I. Wiley-Interscience, 1958.

[41] Easter, M., Kriegel, H. P., Sanders, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD 1996)* (1996), pp. 226–231.

[42] El-Hamdouchi, A., and Willett, P. Hierarchic document classification using ward's clustering method. In *Proceedings of the 9th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 1986)* (New York, NY, USA, 1986), pp. 149–156.

[43] Elkan, C. Using the triangle inequality to accelerate k-means. In *Proceedings of the 20th International Conference on Machine Learning (ICML 2003)* (Washington DC, USA, 2003), pp. 147–153.

[44] Eppstein, D. Fast hierarchical clustering and other applications of dynamic closest pair. *The ACM Journal of Experimental Algorithmics 5*, 1 (June 2000), 1–23.

[45] Eppstein, D. Squarepants in a tree: sum of subtree clustering and hyperbolic pants decomposition. In *Proceedings of the 18th annual ACM-SIAM symposium on Discrete algorithms (SODA 2007)* (Philadelphia, PA, USA, 2007), pp. 29–38.

[46] Eppstein, D., Paterson, M. S., and Yao, F. F. On nearest-neighbor graphs. *Discrete and Computational Geometry 17* (1997), 263–282.

[47] Equitz, W. H. A new vector quantization clustering algorithm. *IEEE Transactions on Acoustics, Speech and Signal processing 37*, 10 (October 1989), 1568–1575.

[48] Farrel, K., Mammone, R., and Assaleh, K. Speaker recognition using neural networks and conventional classifiers. *IEEE Transactions on Speech and Audio Processing 2*, 1 (1994), 194–205.

[49] Feldman, D., Monemizadeh, M., and Sohler, C. A PTAS for k-means clustering based on weak coresets. In *Proceedings of the 23rd annual symposium on Computational geometry (SCG 2007)* (New York, NY, USA, 2007),

ACM, pp. 11–18.

[50] Fessler, J. A., and Hero, A. O. Space-alternating generalized expectation-maximization algorithm. *IEEE Transactions on Signal Processing 42*, 10 (October 1994), 2664–2677.

[51] Figueiredo, M., and Jain, A. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 3 (March 2002), 1–16.

[52] Fränti, P., Kaukoranta, T., Shen, D. F., and Chang, K.-S. Fast and memory efficient implementation of the exact pnn. *IEEE Transactions on Image Processing 9*, 5 (2000), 358–369.

[53] Fränti, P., and Kivijärvi, J. Randomized local search algorithm for the clustering problem. *Pattern Analysis & Applications 3*, 4 (2000), 358–369.

[54] Fränti, P., Virmajoki, O., and Hautamäki, V. Fast agglomerative clustering using a k-nearest neighbour graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 11 (November 2006), 1875–1881.

[55] Fränti, P., Virmajoki, O., and Kaukoranta, T. Branch-and-bound technique for solving optimal clustering. In *Proceedings of the 16th International Conference on Pattern Recogntion (ICPR 2002)* (Quebec, Canada, August 2002), pp. 232–235.

[56] Freedman, D., Purves, R., and Pisani, R. *Statistics*. W.W. Norton, New York, 1978.

[57] Friedman, J. H., Bentley, J. L., and Finkel, R. A. An algorithm for finding best matches in logarithmic exptected time. *ACM Transactions on Mathematical Software 3*, 3 (1977), 209–226.

[58] Garey, M. R., and Johnson, D. S. *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman, San Francisco, 1979.

[59] Gauvain, J. L., and Lee, C.-H. Maximum *a posteriori* estimation for multivariate Gaussian mixture observations of Markov chain. *IEEE Transactions on Speech and Audio Processing 2*, 2 (April 1994), 291–298.

[60] Gersho, A., and Gray, R. M. *Vector Quantization and Signal Compression*. Kluwer Academic Publisher, Boston, USA, 1992.

[61] Ghoting, A., and Parthasarathy, S. Knowledge-conscious exploratory data clustering. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2006)* (Berlin, Germany, September 2006), pp. 511–519.

[62] Gower, J. C., and Ross, G. I. Maximum spanning trees and single linkage cluster analysis. *Applied Statistics 18*, 1 (1969), 54–64.

[63] Guha, S., Rastogi, R., and Shim, K. CURE an efficient clustering algorithm for large databases. In *Proceedings of the 17th International Conference on Management of Data (SIGMOD 1998)* (Seattle, Washington, June 1998), pp. 73–84.

[64] Guha, S., Rastogi, R., and Shim, K. ROCK: A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering (ICDE 1999)* (1999), pp. 512–521.

[65] Hansen, P., and Delattre, M. Complete-link cluster analysis by graph coloring. *Journal of American Statistical Association 73*, 362 (June 1978), 397–403.

[66] Hansen, P., and Jaumard, B. Cluster analysis and mathematical programming. *Mathematical Programming 79*, 1–3 (October 1997), 191–215.

[67] Har-Peled, S., and Sadri, B. How fast is the k-means method? *Algorithmica 41* (2005), 185–202.

[68] Harmegnies, B. SDDD: A new dissimilarity index for the comparison of speech spectra. *Pattern Recognition Letters 8*, 3 (October 1988), 153–158.

[69] Harrington, J., and Cassidy, S. *Techniques in Speech Acoustics.* Kluwer Academic Publishers, Dordrecth, 1999.

[70] Hartigan, J. A. *Clustering Algorithms.* Wiley, New York, 1975.

[71] Hautamäki, V., Cherednichenko, S., Kärkkäinen, I., Kinnunen, T., and Fränti, P. Improving k-means by outlier removal. In *Proceedings of the 14th Scandinavian Conference on Image Analysis (SCIA 2005)* (Joensuu, Finland, June 2005), pp. 978–987.

[72] Hautamäki, V., Kärkkäinen, I., and Fränti, P. Outlier detection using k-nearest neighbour graph. In *International Conference on Pattern Recognition (ICPR 2004)* (Cambridge, UK, August 2004), vol. 3, pp. 430–433.

[73] Hautamäki, V., Kinnunen, T., and Fränti, P. Text-independent speaker recognition using graph matching. *Pattern Recogntion Letters 29*, 9 (July 2008), 1427–1432.

[74] Hautamäki, V., Kinnunen, T., Kärkkäinen, I., Saastamoinen, J., Tuononen, M., and Fränti, P. Maximum *a posteriori* adaptation of the centroid model for speaker verification. *IEEE Signal Processing Letters 15* (2008), 162–165.

[75] Hautamäki, V., Tuononen, M., Niemi-Laitinen, T., and Fränti, P. Improving speaker verification by periodicity based voice activity detection. In *Proceedings of the 12th International Conference on Speech and Computer (SPECOM 2007)* (Moscow, Russia, October 2007), vol. 2, pp. 645–650.

[76] Hawkins, D. M. *Identification of Outliers.* Chapman and Hall, London, 1980.

[77] He, Y., and Chen, L. MinClue: A MST-based clustering method with auto-threshold-detection. In *IEEE Conference on Cybernetics and Intelligent Systems* (Singapore, December 2004), pp. 229–233.

[78] Hochbaum, D. S. When are NP-hard location problems easy? *Annals of Operations Research 1* (1984), 201–214.

[79] Hollien, H., and Majewski, W. Speaker identification by long-term spec-

tra under normal and distorted speech conditions. *Journal of the Acoustical Society of America 62*, 4 (October 1977), 975–980.

[80] Huijbregts, M., and Wooters, C. The blame game: Performance analysis of speaker diarization system components. In *Proceedings of 8th International Interspeech Event (Interspeech 2007)* (August 2007), pp. 1857–1860.

[81] Hwang, F. K., Onn, S., and Rothblum, U. G. A polynomial time algorithm for shaped partition problems. *SIAM Journal of Optimization 10* (1999), 70–81.

[82] Inaba, M., Katoh, N., and Imai, H. Applications of weighted voronoi diagrams and randomization to variance-based k-clustering. In *Proceedings of the 10th Annual ACM symposium on computational geometry (SCG 1994)* (1994), pp. 332–339.

[83] Jain, A. K., and Dubes, R. C. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[84] Jarvis, R. A., and Patrick, E. A. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers C-22*, 11 (November 1973), 1025–1034.

[85] Jensen, R. A dynamic programmig algorithm for cluster analysis. *Operations Research 17*, 6 (November–December 1969), 1034–1057.

[86] Jiang, M. F., Tseng, S. S., and Su, C. M. Two-phase clustering process for outliers detection. *Pattern Recognition Letters 22* (2001), 691–700.

[87] Jin, W., Tung, A. K. H., and Han, J. Finding top-n local outliers in large database. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2001)* (2001), pp. 293–298.

[88] Johnson, S. C. Hierarchical clustering schemes. *Psychometrika 32*, 3 (September 1979), 241–254.

[89] Kang, S.-I., Jo, Q.-H., and Chang, J.-H. Discriminative weight training for statistical model-based voice activity detection. *IEEE Signal Processing Letters 15* (2008), 170–173.

[90] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, W. Y. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24* (2002), 881–892.

[91] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, W. Y. A local search approximation algorithm for k-means clustering. *Computational Geometry 28* (2004), 89–112.

[92] Karypsis, G., Han, E.-H., and Kumar, V. CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer 32*, 8 (August 1999), 68–75.

[93] Kaufman, L., and Rousseeuw, P. J. *Finding groups in data: an introduction to cluster analysis.* John Wiley Sons, New York, 1990.

[94] Kaukoranta, T., Fränti, P., and Nevalainen, O. Vector quantization by lazy pairwise nearest neighbor method. *Optical Engineering 38*, 11 (November 1999), 1862–1868.

[95] Kaukoranta, T., Fränti, P., and Nevalainen, O. A fast GLA based on code vector activity detection. *IEEE Transactions on Image Processing 9*, 8 (August 2000), 1337–1342.

[96] Kenny, P., Boulianne, G., Ouellet, P., and Dumouchel, P. Speaker and session variability in GMM-based speaker verification. *IEEE Transactions on Audio, Speech and Language Processing 15*, 4 (May 2007), 1448–1460.

[97] Kinnunen, T. *Spectral Features for Automatic Text-Independent Speaker Recognition.* Licentiate's thesis, University of Joensuu, Department of Computer Science, Joensuu, Finland, 2004.

[98] Kinnunen, T. *Optimizing Spectral Feature Based Text-independent Speaker Recognition.* PhD thesis, University of Joensuu, 2005.

[99] Kinnunen, T., and González Hautamäki, R. Long-term F0 modeling for text-independent speaker recognition. In *Proceedings of the 10th International Conference Speech and Computer (SPECOM 2005)* (Patras, Greece, October 2005), pp. 567–570.

[100] Kinnunen, T., Hautamäki, V., and Fränti, P. On the use of long-term average spectrum in automatic speaker recogntion. In *Proceedings of the 5th International Symposium on Chinese Spoken Language Processing (ISCSLP 2006)* (Singapore, December 2006), vol. 2, pp. 559–567.

[101] Kinnunen, T., Karpov, E., and Fränti, P. Real-time speaker identification and verification. *IEEE Transactions on Audio, Speech and Language Processing 14*, 1 (January 2006), 277–288.

[102] Kinnunen, T., Kilpeläinen, T., and Fränti, P. Comparison of clustering algorithms in speaker identification. In *Proceedings of the IASTED International Conference on Signal Processing and Communications (SPC 2000)* (Marbella, Spain, September 19-22 2000), pp. 222–227.

[103] Kinnunen, T., Tuononen, M., and Fränti, P. Which clustering algorithm to select for text-independent speaker recognition? *Pattern Recognition* (2008). Submitted.

[104] Kinoshita, Y., Ishihara, S., and Rose, P. Beyond the long-term mean: Exploring the potential of F0 distribution parameters in traditional forensic speaker recognition. In *Proceesings of the Odyssey: The Speaker and Language Recognition Workshop* (Stellenbosch, South Africa, January 2008).

[105] Knorr, E. M., and Ng, R. T. Unified notion of outliers: Properties and computation. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD 1997)* (Newport Beach, CA, 1997), pp. 219–222.

[106] Knorr, E. M., and Ng, R. T. Algorithms for mining distance-based outliers

in large datasets. In *Proceedings 24th International Conference Very Large Data Bases (VLDB 1998)* (New York, USA, 1998), pp. 392–403.

[107] Koontz, W. L. G., Nareda, P. M., and Fukunaga, K. A branch and bound clustering algorithm. *IEEE Transactions on Computers 24*, 9 (September 1975), 908–915.

[108] Kou, Y., Lu, C.-T., and Chen, D. Spatial weighted outlier detection. In *Proceedings of the 5th SIAM International Conference on Data Mining (SDM 2006)* (2006).

[109] Krishnapuram, R., and Keller, J. A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems 1*, 2 (1993), 98–110.

[110] Kuncheva, L. *Fuzzy Classifier Design.* Physica-Verlag, 2000.

[111] Kurita, T. An efficient agglomerative clustering using a heap. *Pattern Recognition 25*, 3 (March 1991), 205–209.

[112] Lee, W., Stolfo, S. J., and Mok, K. W. A data mining framework for building intrusion detection models. In *Proceedings of the IEEE Symposium on Security and Privacy* (May 1999), pp. 120–132.

[113] Lee, Y., Seo, C., Lee, J., and Lee, K. Y. Speaker verification system for PDA in mobile-commerce. In *Proceedings of the 2nd International Conference on Human Society@Internet (HSI 2003)* (Seoul, Korea, June 2003), pp. 668–674.

[114] Liao, Y., and Vemuri, V. R. Use of k-nearest neighbor classifier for intrusion detection. *Computers & Security 21*, 5 (2002), 439–448.

[115] Linde, Y., Buzo, A., and Gray, R. M. An algorithm for vector quantizer desing. *IEEE Transactions 28*, 1 (January 1980), 84–95.

[116] Lindh, J. Visual acoustic vs. aural perceptual speaker identification in closed set of disguised voices. In *Proceedings of The 18th Swedish Phonetics Conference (FONETIK 2005)* (2005), pp. 17–20.

[117] Liu, J. Neural networks with enhanced outlier rejection ability for off-line handwritten word recognition. *Pattern Recognition 35* (2002), 2061–2071.

[118] Liu, J., and Gader, P. Outlier rejection with MLPs and variants of RBF networks. In *Proceedings of The 15th International Conference on Pattern Recognition (ICPR 2000)* (2000), pp. 680–683.

[119] Liu, X., Cheng, G., and Wu, J. X. Analyzing outliers cautiously. *IEEE Transactions on Knowledge and Data Engineering 14*, 2 (March/April 2002), 432–437.

[120] Luoma, O., Tuikkala, J., and Nevalainen, O. Accelerating GLA with an M-tree. In *Proceedings of The Second World Enformatika Conference (WEC 2005)* (Turkey, February 2005), pp. 196–199.

[121] M. R. Garey, D. S. J., and Witsenhausen, H. S. The complexity of the generalized lloyd-max problem. *Transactions on Information Theory 28*, 2 (1982), 255–256.

[122] Ma, J., and Perkins, S. Time-series novelty detection using one-class support vector machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2003)* (October 2003), pp. 1741–1745.

[123] MacQueen, J. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability* (University of California, 1967), vol. I, pp. 281–297.

[124] Mak, M.-W., and Tsang, C.-L. Stochastic feature transformation with divergence-based out-of-handset rejection for robust speaker verification. *EURASIP Journal on Applied Signal Processing 4* (January 2004), 452–465.

[125] Mammone, R. J., Zhang, X., and Ramachandran, R. P. Robust speaker recognition: A feature-based approach. *IEEE Signal Processing Magazine 13*, 5 (September 1996), 58–71.

[126] Markel, J., Oshika, B., and Gray, A. H. Long-term feature averaging for speaker recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing 25*, 4 (1977), 330–337.

[127] Markou, M., and Singh, S. Novelty detection: A review. *Signal Processing 83*, 12 (December 2003), 2481 – 2497.

[128] Martínez, A. M., and Vitriàc, J. Learning mixture models using a genetic version of the EM algorithm. *Pattern Recognition Letters 21*, 8 (July 2000), 759–769.

[129] Matousek, J. On approximate geometric k-clustering. *Discrete Computational Geometry 24* (2000), 61–84.

[130] McKenzie, P., and Alder, M. Initializing the EM algorithm for use in Gaussian mixture modelling. Tech. rep., University of Western Australia, 1993.

[131] Meilă, M. The uniqueness of a good optimum for k-means. In *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)* (Pittsburgh, PA, USA, 2006), pp. 625–632.

[132] Merle, O. D., Hansen, P., Jaumard, B., and Mladenović, N. An interior point algorithm for minimum sum-of-squares clustering. *SIAM Journal of Scientific Computing*, 4 (2000), 1485–1505.

[133] Minigawa, A., Tagawa, N., and Tanaka, T. SMEM algorithm is not fully compatible with maximum-likelihood framework. *Neural Computation 14* (2002), 1261–1266.

[134] Moore, A. The anchors hierarchy: Using the triangle inequality to survive high-dimensional data. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI 2000)* (San Francisco, CA, USA, 2000), pp. 397–405.

[135] Murtagh, F. A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal 26*, 4 (1983), 354–359.

[136] Oppenheim, A., and Schafer, R. *Digital Signal Processing.* Prentice Hall,

1975.

[137] Ostrovsky, R., Rabani, Y., Schulman, L. J., and Swamy, C. The effectiveness of Lloyd-type methods for the k-means problem. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)* (2006), pp. 165–176.

[138] Päivinen, N. Clustering with a minimum spanning tree of scale-free-like structure. *Pattern Recognition Letters 26*, 7 (2005), 921–930.

[139] Papadimitrou, C. Worst-case and probabilistic analysis of a geometric location problems. *SIAM Journal on Computing* (1981).

[140] Patcha, A., and Park, J.-M. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks 51* (2007), 3448–3470.

[141] Pavan, M., and Pelillo, M. Dominant sets and pairwise clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 1 (2007), 167–172.

[142] Peña, J. M., Lozano, J. A., and Larrañaga, P. An empirical comparison of four inititialization methods for the k-means algorithm. *Pattern Recognition Letters*, 10 (October 1999), 1027–1040.

[143] Pelecanos, J., and Sridharan, S. Feature warping for robust speaker verification. In *Proceedings of the Speaker Odyssey 2001* (Crete, Greece, 2001), pp. 213–218.

[144] Pelleg, D., and Moore, A. W. Accelerating exact k-means algorithms with geometric reasoning. In *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining (KDD 1999)* (San Diego, California, USA, 1999), pp. 277–281.

[145] Peng, J., and Xia, Y. A cutting algorithm for the minimum sum-of-squared error clustering. In *Proceedings of the fifth SIAM International Conference on Data Mining (SDM 2005)* (Newport Beach, California, April 2005), pp. 150–160.

[146] Pernkopf, F., and Bouchffra, D. Genetic-based EM algorithm for learning Gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 8 (2005), 1344–1348.

[147] Ramaswamy, S., Rastogi, R., and Shim, K. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 19th International Conference on Management of Data (SIGMOD 2000)* (Dallas, Texas, May 2000), pp. 427–438.

[148] Ramírez, J., Segura, J., Benítez, C., de la Torre, A., and Rubio, A. Efficient voice activity detection algorithms using long-term speech information. *Speech Communication 42* (2004), 271–287.

[149] Rand, W. M. Objective criteria for the evaluation of clustering methods. *Journal of American Statistical Association 66*, 336 (December 1971), 846–

850.

[150] Rao, M. R. Cluster analysis and mathematical programming. *Journal of the American Statistical Association 66*, 335 (September 1971), 622–626.

[151] Reynolds, D., Andrews, W., Campbell, J., Navratil, J., Peskin, B., Adami, A., Jin, Q., Klusacek, D., Abramson, J., J., M. R. G., Jones, D., and Xiang, B. The SuperSID project: exploiting high-level information for high-accuracy speaker recognition. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP 2003)* (Hong Kong, 2003), pp. 784–787.

[152] Reynolds, D., Quatieri, T., and Dunn, R. Speaker verification using adapted gaussian mixture models. *Digital Signal Processing 10*, 1 (2000), 19–41.

[153] Reynolds, D., and Rose, R. Robust text-independent speaker identification using Gaussian mixture models. *IEEE Transactions on Speech and Audio Processing 3* (1995), 72–83.

[154] Rodríguez Liñares, L., Gracía Mateo, C., and Alba Castro, J. On combining classifiers for speaker authentication. *Pattern Recognition 36* (2003), 347–359.

[155] Rose, P. *Forensic Speaker Identification.* Taylor & Francis, London, 2002.

[156] Sambur, M. Selection of acoustic features for speaker identification. *IEEE trans. on Acoustics, Speech, and Singal Processing 23*, 2 (April 1975), 176–182.

[157] Schaeffer, S. E. Graph clustering. *Computer Science Review 1*, 1 (August 2007), 27–64.

[158] Shanbehzadeh, J., and Ogunbona, P. O. On the computational complexity of the LBG and PNN algorithms. *IEEE Transactions on Image Processing 6* (1997), 614–616.

[159] Siohan, O., and Lee, C.-H. Iterative noise and channel estimation under the stochastic matching algortihm framework. *IEEE Signal Processing Letters 4*, 11 (November 1997), 304–306.

[160] Soong, F., Rosenberg, A., Juang, B.-H., and Rabiner, L. R. A vector quantization approach to speaker recognition. *AT & T Technical Journal 66* (1987), 14–26.

[161] Späth, H. *Cluster analysis algorithms for data reduction and classification of objects.* Wiley, new York, 1980.

[162] Sperotto, A., and Pelillo, M. Szemerédi regularity lemma and its applications to pairwise clustering and segmentation. In *The 6th International Conference Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR 2007)* (EZhou, Hubei, China, August 2007), pp. 13–27.

[163] Srebro, N. Are there local maxima in the infinite-sample likelihood of Gaussian mixture estimation? In *Proceedings of the 20th Annual Conference on*

Learning Theory (COLT 2007) (San Diego, California, June 2007), pp. 628–629.

[164] Steen, L. A., and Seebach, A. *Counterexamples in Topology.* Dover publications, 1995.

[165] Steinley, D. k-means clustering: A half-century synthesis. *British journal of Mathematical and Statistical Psychology 59* (2006), 1–34.

[166] Tawfick, M. M., Abbas, H. M., and Shahein, H. I. An integer-coded evolutionary approach for mixture maximum likelihood clustering. *Pattern Recogntion Letters 29* (2008), 515–524.

[167] Tax, D. M. J., and Duin, R. P. W. Support vector data description. *Pattern Recognition Letters 20* (1999), 1191–1199.

[168] Teunen, R., Shahshahani, B., and Heck, L. A model-based transformational approach to robust speaker recognition. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP 2000)* (Beijing, China, 2000), vol. 2, pp. 495–498.

[169] Toledo-Ronen, O. Speech detection for text-dependent speaker verification. In *Proceedings of the Speaker Odyssey 2001* (Crete, Greece, June 2001), pp. 33–36.

[170] Tong, R., Ma, B., Lee, K., You, C., Zhou, D., Kinnunen, T., Sun, H., Dong, M., Chng, E., and Li, H. Fusion of acoustic and tokenization features for speaker recognition. In *Proceedings of the 5th International Symposium on Chinese Spoken Language Processing (ISCSLP 2006)* (Singapore, December 2006), pp. 494–505.

[171] Tranter, S. E., and Reynolds, D. A. An overview of automatic speaker diarization systems. *IEEE Transactions on Audio, Speech, and Language Processing 14*, 5 (September 2006), 1557–1565.

[172] Tuononen, M., González Hautamäki, R., and Fränti, P. Automatic voice activity detection in different speech applications. In *Proceedings of the International Conference on Forensic Applications and Techniques in Telecommunications, Information and Multimedia (e-Forensics 2008)* (Adelaide, Australia, January 2008).

[173] Ueda, N., Nakano, R., Ghahramani, Z., and Hinton, G. E. SMEM algorithm for mixture models. *Neural Computation 12* (2000), 2109–2128.

[174] Vaidya, P. An $o(n \log n)$ algorithm for the all-nearest-neighbors problem. *Discrete and Computational Geometry 4* (1989), 101–115.

[175] Varošanec-Škarić, G., and Bićanić, J. A comparison of indices of difference and similarity, based on LTASS and tested on voices in real forensic case and in controlled conditions. In *Proceedings of the 16th International Congress of Phonetic Sciences (ICPhS 2007)* (Saarbücken, Germany, August 2007), pp. 2085–2088.

[176] Vazirani, V. *Approximation Algorithms.* Springer, 2003.

[177] Velho, L., Gomez, J., Vicinius, M., and Sobreiro, R. Color image quantization by pairwise clustering. In *Proceedings of the Brazilian Symposium of Computer Graphics and Image Processing (SIBGRAPI 1997)* (Campos de Jordão, Brazil, October 1997), pp. 596–601.

[178] Vempala, S., and Wang, G. A spectral algorithm for learning mixture models. *Journal of Computer and System Sciences 68*, 4 (June 2003), 841–860.

[179] Verbeek, J. J., Vlassis, N., and Krose, B. Efficient greedy learning of Gaussian mixture models. *Neural Computation 15*, 2 (2003), 129–151.

[180] Virmajoki, O., Fränti, P., and Kaukoranta, T. Practical methods for speeding-up the pairwise nearest neighbor method. *Optical Engineering 40*, 11 (November 2001), 2495–2504.

[181] Vogt, R., and Sridharan, S. Explicit modeling of session variability for speaker verification. *Computer Speech & Language 22*, 1 (January 2008), 17–38.

[182] Ward, J. H. A new vector quantization clustering algorithm. *Journal of American Statistical Association 58* (1963), 236–244.

[183] Welch, P. D. The use of fast Fourier transforms for the estimation of the power spectra. *IEEE Transactions on Audio and Electroacoustics 15* (1967), 70–73.

[184] Williams, G., Baxter, R., He, H., Hawkings, S., and Gu, L. A comparative study of RNN for outlier detection in data mining. In *Proceedings of the 2nd IEEE International Conference on Data Mining (ICDM 2002)* (Maebashi City, Japan, December 2002).

[185] Xu, R., and Wunsch, D. Survey of clustering algorithms. *IEEE Transactions on Neural Networks 16*, 3 (May 2005), 645–678.

[186] Xu, Y., Olman, V., and Xu, D. Minimum spanning trees for gene expression data clustering. *Genome Informatics 12* (2001), 24–33.

[187] Yamanishi, K., Takeuchi, J., Williams, G., and Milne, P. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithm. In *Proceedings The 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2000)* (August 2000), pp. 320–324.

[188] Yin, J., Yang, Q., and jeffrey Junfeng Pan. Sensor-based abnormal human-activity detection. *IEEE Transactions on Knowledge and Data Engineering* (2008). (in press).

[189] Zhang, T., Ramakrishnan, R., and Livny, M. BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery 1*, 2 (1997), 141–182.

[190] Zhang, Y., yang, S., and Wang, Y. Y. LDBOD: A novel local distribution based outlier detector. *Pattern Recogntion Letters 29* (May 2008), 967–976.

[191] Zhang, Z., Dai, B. T., and Tung, A. K. H. On the lower bound of local optimums in k-means algorithm. In *Proceedings of the 6th International Conference on Data Mining (ICDM 2006)* (Los Alamitos, CA, USA, 2006), pp. 775–786.

[192] Äyrämö, S. *Knowledge Mining Using Robust Clustering.* PhD thesis, University of Jyväskylä, 2006.

1

Publication P1

P. Fränti, O. Virmajoki and V. Hautamäki, "Fast agglomerative clustering using a *k*-nearest neighbor graph", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, No. 11, pp. 1875-1881, November 2006.

Copyright © 2006 IEEE. Reprinted with permission.

Fast Agglomerative Clustering Using a k-Nearest Neighbor Graph

Pasi Fränti, Olli Virmajoki, and Ville Hautamäki

Abstract—We propose a fast agglomerative clustering method using an approximate nearest neighbor graph for reducing the number of distance calculations. The time complexity of the algorithm is improved from $O(\tau N^2)$ to $O(\tau N \log N)$ at the cost of a slight increase in distortion; here, τ denotes the number of nearest neighbor updates required at each iteration. According to the experiments, a relatively small neighborhood size is sufficient to maintain the quality close to that of the full search.

Index Terms—Clustering, agglomeration, nearest neighbor, vector quantization, PNN.

1 Introduction

The *agglomerative clustering* method [1] is commonly used for clustering because of its conceptual simplicity. Its main drawback is its slowness, as the original implementation requires $O(N^3)$ time [2]. An order of magnitude faster algorithm has been introduced in [3], but the method is still lower bounded by $\Omega(N^2)$. The main source of computation originates from the search of the nearest neighbor.

Another approach is to use graph theoretical methods [4], [5]. For example, by first creating a complete undirected graph, where the nodes correspond to the data vectors. The graph can be trimmed to a *minimal spanning tree* representing one large cluster, which is then iteratively split by removing largest edges one by one. The final clustering is then determined by finding the separated components [4]. This is a variation of divisive clustering with similar criteria as in the *single-linkage* agglomerative clustering [6]. For agglomerative clustering, graph theoretical methods have also been used by constructing a sparse graph and then performing the clustering on this graph [7], [8].

We introduce a fast agglomerative clustering algorithm motivated by the graph-based approaches. However, we process the data so that every node represents a cluster and not a single vector. The main difference is that the existing methods neglect the original data after building the weighted graph, whereas we use the original data in order to compute the weights of newly formed edges as the agglomerative clustering goes on. We use the graph merely as a search structure for reducing the number of distance calculations.

The proposed approach has two specific problems: how to generate the graph efficiently, and how to utilize it. For example, standard solutions for solving the minimum spanning tree take $O(N^2)$ time, which would overweigh any speedup. For the first problem, we consider *K-d tree* [9], *divide-and-conquer* [10], and *projection-based search* [11]. For the second problem, we consider a double linked list for utilizing the graph structure. Experiments show that a significant speedup can be achieved and a relatively small neighborhood size is sufficient for preserving the quality of the clustering.

The rest of the paper is organized as follows: In Section 2, we define the clustering problem and recall the agglomerative algorithm. In Section 3, we propose the new graph-based agglomerative clustering algorithm. Solutions for creating the nearest neighbor graph are considered in Section 4. Experimental results are reported in Section 5 and conclusions are drawn in Section 6.

2 Agglomerative Clustering

The *clustering problem* is defined here as a combinatorial optimization problem. Given a set of N data vectors $X = \{x_1, x_2, \ldots, x_N\}$, partition the data set into M clusters so that a given distortion function is minimized. Partition $P = \{p_1, p_2, \ldots, p_N\}$ defines the clustering by giving for each data vector the index of the cluster where it is assigned to. *A cluster* s_a is defined as the set of data vectors that belong to the same partition a:

$$s_a = \{x_i | p_i = a\}. \tag{1}$$

The clustering is then represented as the set $S = \{s_1, s_2, \ldots, s_M\}$. In vector quantization, the output of the clustering is a codebook $C = \{c_1, c_2, \ldots, c_M\}$, which is usually the set of cluster centroids. We assume that the vectors belong to Euclidean space and use the mean square error (MSE) as the distortion function:

$$MSE(C, P) = \frac{1}{N} \cdot \sum_{i=1}^{N} \left\| x_i - c_{p_i} \right\|^2. \tag{2}$$

The *agglomerative clustering* method [1], [12] generates the clustering hierarchically using a sequence of merge operations. At each iteration, two nearby clusters are merged:

$$s_a \leftarrow s_a \cup s_b. \tag{3}$$

The cost of merging two clusters s_a and s_b is the increase in the *MSE*-value caused by the merge. It can be calculated using the following formula [1], [12]:

$$MergeCost(a, b) = \frac{n_a n_b}{n_a + n_b} \cdot \|c_a - c_b\|^2, \tag{4}$$

where n_a and n_b are the corresponding cluster sizes. This will be later denoted as the distance of the clusters a and b. *Ward's method* [1] selects the cluster pair to be merged that minimizes the increase in the distortion function value:

$$a, b = \arg\min_{\substack{i,j \in [1,m] \\ i \neq j}} MergeCost(i, j), \tag{5}$$

where m is the current number of clusters. In the vector quantization context, this is known as the *pairwise nearest neighbor (PNN)* method due to [12]. Straightforward implementation recalculates all distances at each iteration of the algorithm. This takes $O(N^3)$ time because there are $O(N)$ iterations, and $O(N^2)$ cluster pairs to be checked at each iteration.

Another approach is to maintain an $N \times N$ matrix of the merge cost values. The merge cost values must be updated only for the newly merged cluster. Nevertheless, the algorithm still requires $O(N^3)$ because the search of the minimum cluster pair takes $O(N^2)$ time [2]. Kurita's method maintains an $N \times N$ matrix, but it also utilizes a heap structure for searching the minimum distance [13].

• *The authors are with the Speech and Image Processing Unit, Department of Computer Science, University of Joensuu, PO Box 111, FIN-80101 Joensuu, Finland. E-mail: {franti, ovirma, villeh}@cs.joensuu.fi.*

Manuscript received 3 Oct. 2005; revised 23 Mar. 2006; accepted 3 Apr. 2006; published online 14 Sept. 2006.
Recommended for acceptance by L. Kuncheva.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0534-1005.

0162-8828/06/$20.00 © 2006 IEEE Published by the IEEE Computer Society

AgglomerativeClustering(X, M) $\rightarrow S$

Step 1: Construct k-NN graph.
Step 2: REPEAT
Step 2.1: Find the best pair (s_a, s_b) to be merged.
Step 2.2: Merge the pair (s_a, s_b) $\rightarrow s_{ab}$.
Resolve k-NN list for s_{ab}.
Step 2.3: Remove obsolete (s_b).
Step 2.4: Find neighbors of s_{ab}.
Step 2.5: Update distances for the neighbors of s_{ab}.
UNTIL $|S|=M$;

Fig. 1. The structure of the proposed algorithm.

The method runs in $\mathrm{O}(N^2 \cdot \log N)$ time, but the matrix requires $\mathrm{O}(N^2)$ memory.

A fast implementation with linear memory consumption maintains a pointer from each cluster to its nearest neighbor [3]. The cluster pair to be merged can be found in $\mathrm{O}(N)$ time and only a small number (denoted by τ) of the nearest neighbors need to be updated after each merge. The implementation takes $\mathrm{O}(\tau N^2)$ time in total. We refer to this as the *fast exact PNN*.

Further speedup can be achieved by using lazy update of the merge cost values [14] and by reducing the amount of work caused by the distance calculations [15]. It has been shown that in a one-dimensional case (multilevel thresholding), agglomerative clustering can be performed in $\mathrm{O}(N \cdot \log N)$ time [16]. The result, however, does not generalize to higher dimensional data.

An inexact $\mathrm{O}(N \cdot \log N)$ time variant has also been considered in [12] by using *K-d* trees for localizing the search for the clusters and by merging several cluster pairs during the same iteration. This variant can be very fast, but it significantly decreases the quality of the clustering. Nevertheless, the idea itself of limiting the search of the nearest cluster is appropriate, and we continue the work in the same direction.

3 Agglomerative Clustering Using k-NN Graphs

We define a *k-nearest neighbor graph (k-NN graph)* as a weighted directed graph, in which every node represents a single cluster and the edges represent pointers to neighbor clusters. Every node has exactly k edges to the k nearest clusters, according to (4).

In agglomerative clustering, the search for the nearest neighbor is repeated several times per iteration and every search requires $\mathrm{O}(N)$ merge cost calculations. The graph is utilized so that the search is limited only to the clusters that are directly connected by the graph structure. This reduces the time complexity of every search from $\mathrm{O}(N)$ to $\mathrm{O}(k)$. The parameter k affects the quality of the solution and the running time. If the number of neighbors (k) is small, significant speedup can be obtained.

3.1 The Simple Algorithm

The main structure of the algorithm is given in Fig. 1. The algorithm starts by constructing the neighborhood graph in the first step and, then, iteratively merges pairs of clusters until the desired number of clusters has been reached.

At Step 2.1, the edge with the smallest weight is found. For each cluster, we store the smallest merge cost value associated to the edge in a *heap structure* [17]. It is a binary tree, in which the minimum value is stored in the root (top of heap) and the values of the children are always greater than that of the parent node. In this way, we always find the best pair to be merged from the top of the heap and the heap structure can be updated in logarithmic time after the changes in the merge cost values. Note that only the nearest of the k-NN pointers is stored for each cluster.

Fig. 2. An illustration of the k-NN graph ($k = 2$) where a and b are to be merged.

At Step 2.2, the nodes (s_a and s_b) are merged and a new k-NN list is constructed for s_{ab}. In order to keep the computation reasonable, we select the k nearest neighbors from the $2k$ neighbors of the previously merged nodes s_a and s_b. This also means that the accuracy of the k-NN graph is compromised and, thus, the graph becomes an approximated nearest neighbor graph. It may also happen that the number of neighbors for the cluster s_{ab} can become smaller than k. The merged node (s_{ab}) replaces s_a and the second cluster (s_b) is removed from the data structures at step 2.3. At Step 2.4, we find the neighbors that pointed to s_a or s_b, and we update the corresponding merge cost values at Step 2.5.

We illustrate the procedure in Fig. 2 for a sample directed 2-NN graph ($k = 2$), where a and b are merged. The new k-NN list of the merged cluster is found among the neighbors of a and b, which are c and e. We also update the edges that pointed to the clusters a and b to point to the new cluster and update the associated cost values. The corresponding locations in the heap structure must also be updated. The new cluster replaces a, and b is removed. The pointers $c \rightarrow b$ and $d \rightarrow b$ are replaced by pointers $c \rightarrow a$ and $d \rightarrow a$, accordingly. A sample graph ($k = 4$) is shown in Fig. 3.

3.2 Time Complexity

The summary of the time complexities are presented in Table 1, where the "steps" indicate the number of times the loops are performed per iteration and the "distances" indicate the number of distance calculations (4). We can consider k as a small constant, whereas in the extreme case ($k = N$), the algorithm would produce exactly the same result as the full search.

At each iteration, the best pair can be found from the heap in $\mathrm{O}(1)$ time (*Find the best pair*). The merge takes $\mathrm{O}(k^2 + \log N)$ time (*Merge*). The first term (k^2) comes from finding the k neighbors from the k-NN lists of a and b. In principle, we could merge the two lists in $\mathrm{O}(k)$ time, but as the merge changes the weights, the lists have no proper ordering anymore after merging and we apply insertion sort requiring $\mathrm{O}(k^2)$ time. The second term ($\log N$) comes from the update of the heap structure.

We then find all the clusters that have the merged one in their k-NN list (*Find neighbors*). This takes $\mathrm{O}(kN)$ time because the only

Fig. 3. An example of the graph with a synthetic data set and the corresponding k-NN graph ($k = 4$). Darker dots illustrate cluster centroids. Arrowheads are not printed for clarity.

TABLE 1
The Estimated Number of Steps and Distance Calculations per Iteration

	Fast exact PNN		Proposed (simple)		Proposed (double-linked)	
	Steps	Distances	Steps	Distances	Steps	Distances
Find the best pair	N	-	1	-	1	-
Merge	N	-	$k^2 + \log N$	k	$k^2 + \tau k + \log N$	k
Remove obsolete	N	-	$k + \log N$	-	$\mathrm{Log}N$	-
Find neighbors	N	-	kN	-	τk	-
Update distances	$N(1+\tau)$	τN	$\tau + \tau/k \cdot \log N$	τ	$\tau + \tau/k \cdot \log N$	τ
Total	$O(\tau N^2)$	$O(\tau N^2)$	$O(kN^2)$	$O(\tau N)$	$O(\tau N \cdot \log N)$	$O(\tau N)$

way to find the neighbors is to browse through the k-NN lists of all N clusters. One of the merged clusters (cluster b in our case) must also be removed (*Remove obsolete*), which takes $O(k + \log N)$ time due to the removal of k pointers and due to the removal of one value from the heap.

Finally, we recalculate the distance values of the neighbor clusters (*Update distances*). This takes $O(\tau + (\tau/k) \cdot \log N)$ time, where τ is the in-degree of the node. There are τ distance values to be updated, each taking only $O(1)$. However, only the first entry from each k-NN list is stored in the heap. We, therefore, approximate that only every kth of the distance update generates heap update. Thus, there are τ/k heap updates on average, each requiring $O(\log N)$ time.

The number of τ depends both on k and on the data. Every cluster has exactly k outgoing pointers and the in-degree of a randomly chosen node must therefore also be k, on average. As there are two clusters involved in the merge, $\tau = 2k$ for a randomly chosen cluster pair. However, the pair to be merged is more likely to be selected from a dense area. Thus, the number of neighbors depends also on the dimensionality and structure of the data, but in a way that is not trivial to measure.It has been shown in [18] that, in Euclidean space, τ is upper bounded by k times the *kissing number*, which is defined as the number of unit hyperspheres that are touching another unit hypersphere without any intersections. The kissing numbers are known for some dimensions, but, in general, the problem is unsolved [19]. It was observed in [3] that there are only a relatively few (about 2-4) neighbors, on average.

3.3 Double Linked Algorithm (DLA)

Even though the number of distance calculations has been greatly reduced by the simple algorithm, the number of steps still sums up to $O(kN^2)$; see Table 1. The bottleneck is the search for the incoming edges, i.e., the nodes that consider the merged cluster as their neighbor. This step dominates the time complexity requiring $O(kN)$ per iteration, thus summing up to $O(kN^2)$ in total.

To attack this problem, we implement a double linked list structure as shown in Fig. 4. For every node, we maintain two lists: the k-NN list containing the pointers to the k nearest neighbors and another containing the "back pointers" to the clusters that consider the particular cluster as part of their k nearest neighbors. For example, in Fig. 4, there are seven clusters that consider either a or b as their nearest neighbors. All of them appear in the back pointer lists of a (h, j, k, b) or b (c, d, a). The differences between the simple algorithm and the double link algorithm (DLA) are described next.

In the merge step, in addition to resolving the k-NN list, we must also update the back pointers of the neighbors. This is done by browsing through the back pointer lists of the neighbors. As there are k neighbors to be checked (clusters c and a in Fig. 4) and the number of back pointers is τ, an additional $O(\tau k)$ term appears in the time complexity of this step. On the other hand, the back pointer lists eliminate the need for the $O(kN)$ time loops in the search of the neighbors and replace it by searching through the back pointer list of the cluster, which takes $O(\tau k)$ time. In Fig. 4, we need to consider the neighbors h, j, k, c, and d. The rest of the algorithm is the same and also the number of distance calculations remains the same.

In the double linked algorithm, there is no clear bottleneck anymore. If we consider k as a constant, we can simplify the overall time complexity to $O(\tau N \cdot \log N)$ originating from the heap updates ($\tau/k \cdot \log N$); see Table 1. In practice, τ is also very small for realistic data sets and, therefore, the merge step takes more

Fig. 4. Illustration of the double linked list structure for a part of the graph presented in Fig. 2 and the required updates due to the merge of clusters a and b.

TABLE 2
The Observed Cumulative Number of Steps and Distance Calculations ($k = 5$) for *Bridge*

	Fast exact PNN		Proposed (simple)		Proposed (double-linked)	
	Steps	Distances	Steps	Distances	Steps	Distances
Find the best pair	8 357 760	-	3 840	-	3 840	-
Merge	8 367 905	-	200 144	22 613	305 326	22 493
Remove obsolete	8 349 185	-	102 450	-	45 514	-
Find neighbors	8 357 760	-	41 769 600	-	204 689	-
Update distances	48 538 136	40 166 328	198 334	57 800	187 123	46 566
Total	81 970 746	40 166 328	42 274 368	80 413	746 492	69 059

computation because $\tau k \geq \tau/k \cdot \log N \Leftrightarrow k^2 \geq \log N$, which is true for our data sets ($k = 5$ and $N = 1{,}000..100{,}000$).

The observed number of steps and distance calculations are reported in Table 2 for a sample data set (see Section 5). The number of distance calculations is reduced to a fraction of that of the full search (from 40 million to 80,413) by the simple implementation, but the number of steps only to 52 percent of that of the full search (from 81 million to 42 million). The double link algorithm, however, reduces the number of steps to 746,492.

4 Creation of the Neighborhood Graph

The exact k-NN graph can be constructed in $\mathrm{O}(N^2)$ time using *brute force* by considering all pairwise distances. We therefore consider faster alternatives for constructing an approximate k-NN graph by using *K-d tree*, *divide-and-conquer*, and *projection-based search*.

The K-d tree method has been widely used for finding the nearest neighbor [9], [20] and, here, we extended it for finding k nearest neighbors. The method creates a tree-structured partition by recursively splitting the data into two disjoint partitions along one of the major axes. The leaf nodes of the resulting tree form disjoint partitions called *buckets*. The nearest neighbor for any vector can be found among the vectors in the same bucket, or from its sibling nodes. The creation of the tree takes $\mathrm{O}(N \cdot \log N)$ time and each search $\mathrm{O}(\log N)$ time, but only if the dimensionality d is considered as a constant.

The *closest pair problem* is stated as follows: Given N points in d-dimensional space, find the two whose mutual distance is the smallest. The problem can be solved by the divide-and-conquer approach [10] that recursively divides the data set into subsets. We first calculate the principal axis of the data vectors and select a $(d-1)$-dimensional hyperplane perpendicular to the axis, so that it divides the set approximately into two halves. A third subset is generated from vectors that are closer to the dividing *hyperplane H* than its nearest neighbor. The three subproblems are solved recursively and the results are then combined. An upper bound for the time complexity can be estimated by the recurrence

$$T(N) = 3 \cdot T(N/2) + \mathrm{O}(N \cdot d^2),$$

which derives to $\mathrm{O}(d^2 \cdot N^{1.58})$. Lower time complexity could be obtained by dividing the hyperplane by a simpler heuristic or by making tighter bounds for the size of the third subset.

Mean-distance ordered partial search (MPS) calculates projections of the data vectors to the diagonal axis [11] and proceeds to search bidirectionally along the axis, starting from the input vector. Given the distance to the best candidate found so far, other vectors can be excluded from the search if the projection is outside of a given radius. The precondition can be calculated in $\mathrm{O}(1)$ time and, if it holds, the $\mathrm{O}(d)$ time distance calculation can be avoided.

For finding the k nearest vectors, we relax the condition of the graph and find any k neighbors, instead of the nearest ones. We use the MPS method for finding the nearest neighbor, but stop the search when it has been found (*MPS full*) or when a predefined search limit has been reached (*MPS limited*). In addition to this, we maintain an ordered list of the k best candidates found so far. The rest of the neighbors are taken as the k best candidates from the list of the candidates once the first nearest neighbor has been found. It

TABLE 3
Summary of the Data Sets

Data set	Type of data set	Number of data vectors (N)	Number of clusters (M)	Dimension of data vector (d)
Bridge	Gray-scale image	4086	256	16
House	*RGB* image	34112	256	3
Miss America	Residual vectors	6480	256	16
$BIRCH_1$-$BIRCH_3$	Synthetically generated	100000	100	2
*Dim*64-1024	Synthetically generated	1000	256	64 – 1024

Fig. 5. The running time and the quality of the *DLA* as functions of k.

is expected that the rest of the candidates are nearby vectors, although not necessarily the nearest ones.

The main advantage of the method is its simplicity and the main disadvantage is that the worst case time complexity is still $O(N^2)$. For details of the algorithm, see [21]. A better projection axis can be obtained by *principal component analysis* (PCA) [22, p. 10] on the data set and by using the projection to the first principal component. It is expected that the search can be terminated earlier by using the principal axis rather than the diagonal one. However, the calculation of the principal axis takes $O(Nd^2)$ time, which might outweigh the additional speedup in the case of higher dimensional data sets.

TABLE 4
Observed Number of Nearest Neighbor Updates (τ) Required

Data set	τ
Bridge	12.1
House	7.0
Miss America	11.1
$BIRCH_1$	5.1
$BIRCH_2$	5.0
$BIRCH_3$	5.1

Fig. 6. Histograms of the observed τ values.

5 Experiments

We consider three image data sets from [3], three *BIRCH* data sets [23], and five high dimensional data sets *Dim64* to *Dim1024* with higher dimensionality varying from 64 to 1,024. The summary of

TABLE 5
Running Times of the Graph Creation Algorithms for the Image Data Sets ($k = 5$)

Algorithm:	*Bridge*		*House*		*Miss America*	
	Time	MSE	Time	MSE	Time	MSE
Brute force	34	170.25	881	6.28	89	5.40
KD-tree	12	169.83	5	6.33	79	5.40
D-n-C	2	171.58	49	6.37	7	5.44
MPS full	3	170.28	19	6.34	45	5.41
MPS limited	3	170.60	14	6.43	6	5.59
MPS / PCA	10	170.23	37	6.28	58	5.38

Fig. 7. Running times of the graph creation algorithms versus dimensionality (*Dim* data sets).

TABLE 6
Time (Seconds) and Quality (MSE) Comparison of the Methods

Image data sets		*Bridge*		*House*		*Miss America*	
		Time	MSE	Time	MSE	Time	MSE
Original PNN	[12]	>9999	168.92	>9999	6.27	>9999	5.36
K-d tree PNN	[12]	2	199.20	65	7.03	5	6.31
Fast exact PNN	[3]	79	168.92	1574	6.27	229	5.36
+PDS+MPS+Lazy	[15]	9	168.92	190	6.26	106	5.37
DLA	Proposed	3	170.60	14	6.43	6	5.59
K-means	[24]	13	179.87	23	7.81	20	5.96
+PDS+MPS+Activity	[25]	2	179.87	3	7.81	8	5.96
PNN + k-means	Combined	80	165.04	191	6.07	109	5.24
DLA + k-means		4	166.35	15	6.13	9	5.34

Birch data sets		$BIRCH_1$		$BIRCH_2$		$BIRCH_3$	
		Time	MSE	Time	MSE	Time	MSE
Original PNN	[12]	>9999	4.73	>9999	2.28	>9999	1.96
K-d tree PNN	[12]	401	7.09	401	3.86	401	2.41
Fast exact PNN	[3]	>9999	4.73	>9999	2.28	>9999	1.96
+PDS+MPS+Lazy	[15]	2397	4.73	2115	2.28	2316	1.96
DLA	Proposed	40	4.74	16	2.28	34	2.01
K-means	[24]	209	5.52	43	7.99	171	2.53
+PDS+MPS+Activity	[25]	29	5.52	8	7.99	35	2.53
PNN + k-means	Combined	2401	4.64	2116	2.28	2333	1.88
DLA + k-means		44	4.64	17	2.28	51	1.91

the data sets is presented in Table 3. The results are run on a 450 MHz Pentium III personal computer.

In all test sets, a relatively small neighborhood size (k) is sufficient to keep the distortion close to that of the full search. The actual neighborhood size is not so critical, as long as the neighborhood is large enough to keep the vectors within the same cluster connected, thus avoiding isolated subclusters. According to our experiments, this can be reached in all test sets presented here by using $k = 3$ or $k = 4$.

The running time has linear dependency with the parameter k, but the growing rate is small; see Fig. 5. The graph creation is the bottleneck of the algorithm. In the rest of the paper, we fix the neighborhood size at $k = 5$. The number of incoming pointers (τ) depends on the data set and on the value of k. The observed numbers for the data sets have been reported in Table 4, and their distributions in Fig. 6. The results clearly show that the number of τ increases with the dimensionality of the data set.

For the graph creation, we consider the following five algorithms:

- *Brute force*
- *K-d tree*
- *Divide-and-conquer (D-n-C)*
- *Projection-based (MPS)*
- *Projection-based (MPS/PCA)*

The *D-n-C* algorithm works well with some of the image data sets (see Table 5), but, as the results in Fig. 7 demonstrate, increasing dimensionality affects its performance severely because of the calculation of the *PCA*, which takes quadratic time. *K-d tree* works best for *House* because of its low dimensionality. For higher dimensional data sets, *MPS* provides the best overall results (limited to 500 searches) and it is, therefore, chosen for the rest of the experiments.

Finally, the proposed method is compared against the existing *PNN* variants and the *k-means* algorithm in Table 6. The *original PNN* includes both the slow $O(N^3)$ and the faster $O(N \cdot \log N)$ time inexact variant using *K-d tree*. The *Fast exact PNN* has two variants: the one proposed in [3] and an improved variant [15] that uses *PDS*, *MPS*, and *Lazy evaluation* for speedup. The *k-means* results include the original method [24] and a faster variant that uses *PDS*, *MPS* and *activity detection* for speedup [25]. The results of the combined methods (*PNN + k-means*, *DLA + k-means*) are produced by feeding the output of the *PNN* (or *DLA*) as the input to the *k-means*.

The proposed method (*DLA*) provides clustering almost as good as (*Birch* sets) or only slightly worse (image sets) than the *PNN*, but significantly faster. The difference is remarkable, especially with the large data sets (*Birch*). Comparison to *k-means* shows that the proposed method provides significantly better

Fig. 8. Histograms of the MSE-values of 500 runs of the *k means*, *PNN*, and *DLA*.

Fig. 9. Sample clustering of a synthetic data set (a) by *k-means* and (b) by DLA with the remaining k-NN graph.

quality with an algorithm that is competitive also in speed. The difference in quality is demonstrated in Fig. 8 and Fig. 9.

6 Conclusions

A fast agglomerative method has been proposed for clustering using an approximate k-nearest neighbor graph. A relatively small neighborhood size is sufficient to produce clustering with similar quality to that of the full search. There are no theoretical grounds for how to fix the exact neighborhood size optimally, but there is one guideline that should be followed: The connectivity of the vectors within the clusters should be preserved. The bottleneck of the algorithm is the graph creation and it remains a challenge to invent a practical algorithm for creating a reasonably accurate k-NN graph in subquadratic time.

References

[1] J.H. Ward, "Hierarchical Grouping to Optimize an Objective Function," *J. Am. Statisical Assoc.*, vol. 58, pp. 236-244, 1963.
[2] J. Shanbehzadeh and P.O. Ogunbona, "On the Computational Complexity of the LBG and PNN Algorithms," *IEEE Trans. Image Processing*, vol. 6, no. 4, pp. 614-616, Apr. 1997.
[3] P. Fränti, T. Kaukoranta, D.-F. Shen, and K.-S. Chang, "Fast and Memory Efficient Implementation of the Exact PNN," *IEEE Trans. Image Processing*, vol. 9, no. 5, pp. 773-777, May 2000.
[4] J.C. Gover and G.J.S. Ross, "Minimum Spanning Trees and Single Linkage Cluster Analysis," *Applied Statistics*, vol. 18, pp. 54-64, 1969.
[5] S. Bandyopadhyay, "An Automatic Shape Independent Clustering Technique," *Pattern Recognition*, vol. 37, no. 1, pp. 33-45, Jan. 2004.
[6] P.H.A. Sneath, "The Application of Computers to Taxonomy," *J. General Microbiology*, vol. 17, no. 1, pp. 210-226, Aug. 1957.
[7] G. Karypis, E. Han, and V. Kumar, "CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling," *Computer*, vol. 32, no. 8, pp. 66-75, Aug. 1999.
[8] D. Harel and Y. Koren, "Clustering Spatial Data Using Random Walks," *Proc. Seventh ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '01)*, pp. 281-286, Aug. 2001.
[9] J.H. Friedman, J.L. Bentley, and R.A. Finkel, "An Algorithm for Finding Best Matches in Logarithmic Expected Time," *ACM Trans. Math. Software*, vol. 3, no. 3, pp. 209-226, Sept. 1977.
[10] O. Virmajoki and P. Fränti, "Divide-and-Conquer Algorithm for Creating Neighborhood Graph for Clustering," *Proc. Int'l Conf. Pattern Recognition (ICPR '04)*, vol. 1, pp. 264-267, Aug. 2004.
[11] S.-W. Ra and J.K. Kim, "A Fast Mean-Distance-Ordered Partial Codebook Search Algorithm for Image Vector Quantization," *IEEE Trans. Circuits and Systems*, vol. 40, no. 9, pp. 576-579, Sept. 1993.
[12] W.H. Equitz, "A New Vector Quantization Clustering Algorithm," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 37, no. 10, pp. 1568-1575, Oct. 1989.
[13] T. Kurita, "An Efficient Agglomerative Clustering Algorithm Using a Heap," *Pattern Recognition*, vol. 24, no. 3, pp. 205-209, Mar. 1991.
[14] T. Kaukoranta, P. Fränti, and O. Nevalainen, "Vector Quantization by Lazy Pairwise Nearest Neighbor Method," *Optical Eng.*, vol. 38, no. 11, pp. 1862-1868, Nov. 1999.
[15] O. Virmajoki, P. Fränti, and T. Kaukoranta, "Practical Methods for Speeding-Up the Pairwise Nearest Neighbor Method," *Optical Eng.*, vol. 40, no. 11, pp. 2495-2504, Nov. 2001.
[16] O. Virmajoki and P. Fränti, "Fast Pairwise Nearest Neighbor Based Algorithm for Multilevel Thresholding," *J. Electronic Imaging*, vol. 12, no. 4, pp. 648-659, Oct. 2003.
[17] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*. Cambridge, Mass.: MIT Press, 1998.
[18] G.L. Miller, S-H. Teng, W. Thurston, and S.A. Vavaris, "Separators for Sphere-Packings and Nearest Neighbor Graphs," *J. ACM*, vol. 44. no. 1, pp. 1-29, Jan. 1997.
[19] J.H. Conway and N.J.A. Sloane, *Sphere Packings, Lattices and Groups*. New York: Springer-Verlag, 1998.
[20] R. Sproull, "Refinements to Nearest-Neighbor Searching in k-d Trees," *Algorithmica*, vol. 6, pp. 579-589, 1991.
[21] P. Fränti, O. Virmajoki, and V. Hautamäki, "Fast PNN-Based Clustering Using *k* Nearest Neighbor Graph," *Proc. IEEE Int'l Conf. Data Mining (ICDM '03)*, pp. 525-528, Nov. 2003.
[22] *Encyclopedia of Statistical Sciences*, vol. 6, S. Kotz, N.L. Johnson, and C.B. Read, eds. New York: John Wiley Sons, 1985.
[23] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: A New Data Clustering Algorithm and Its Applications," *Data Mining and Knowledge Discovery*, vol. 1, no. 2, pp. 141-182, June 1997.
[24] Y. Linde, A. Buzo, and R.M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. Comm.*, vol. 28, no. 1, pp. 84-95, Jan. 1980.
[25] T. Kaukoranta, P. Fränti, and O. Nevalainen, "A Fast Exact GLA Based on Code Vector Activity Detection," *IEEE Trans. Image Processing*, vol. 9, no. 8, pp. 1337-1342, Aug. 2000.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.

2

Publication P2

V. Hautamäki, I. Kärkkäinen and P. Fränti, "Outlier detection using *k*-nearest neighbour graph", in *Proceedings of the 17th International Conference on Pattern Recognition* (ICPR 2004), Vol. 3, pp. 430-433, Cambridge, UK, August 2004.

Copyright © 2004 IEEE. Reprinted with permission.

Outlier Detection Using k-Nearest Neighbour Graph

Ville Hautamäki, Ismo Kärkkäinen and Pasi Fränti
University of Joensuu, Department of Computer Science
Joensuu, Finland
{villeh, iak, franti}@cs.joensuu.fi

Abstract

We present an Outlier Detection using Indegree Number (ODIN) algorithm that utilizes k-nearest neighbour graph. Improvements to existing kNN distance -based method are also proposed. We compare the methods with real and synthetic datasets. The results show that the proposed method achieves resonable results with synthetic data and outperforms compared methods with real data sets with small number of observations.

1. Introduction

Outlier is defined as an observation that deviates too much from other observations that it arouses suspicions that it was generated by a different mechanism from other observations [6]. *Inlier*, on the other hand, is defined as an observation that is explained by underlying probability density function. In clustering, outliers are considered as noise observations that should be removed in order to make more reliable clustering [5]. In data mining, detection of anomalous patterns in data is more interesting than detecting inlier clusters. For example, a breast cancer detection system might consider inlier observations to represent healthy patient and outlier observation as a patient with breast cancer. Similarly computer security intrusion detection system finds an inlier pattern as representation of normal network behaviour and outliers as possible intrusion attempts [13].

The exact definition of an outlier depends on the context. Definitions fall roughly into five categories [7]: i) *distribution-based*, ii) *depth-based*, iii) *distance-based*, iv) *clustering-based* and v) *density-based*. Distribution-based methods originate from statistics, where observation is considered as an outlier if it deviates too much from underlying distribution. For example, in normal distribution outlier is an observation whose distance from the average observation is three times of the variance [4]. The problem is that in real world cases underlying distribution is usually unknown and cannot be estimated from data without outliers affecting the estimate, thus creating a chicken-egg problem. Distance-based methods [8] define outlier as an observation that is d_{min} distance away from p percentage of observations in the dataset. The problem is then finding appropriate d_{min} and p such that outliers would be correctly detected with a small number of false detections. This process usually needs domain knowledge [8]. In clustering-based methods, outlier is defined to be observation that does not fit to the overall clustering pattern [15].

In density-based methods, outlier is detected from local density of observations. These methods use different density estimation strategies. A low local density on the observation is an indication of a possible outlier. For example, Brito *et al.* [1] proposed a *Mutual k-Nearest Neighbor* (MkNN) graph based approach. MkNN graph is a graph where an edge exists between vectors v_i and v_j if they both belong to each others k-neighbourhood. MkNN graph is undirected and is a special case of *k-Nearest Neighbour* (kNN) graph, in which every node has pointers to its k nearest neighbours. Each connected component is considered as a cluster if, it contains more than one vector and an outlier, when connected component contains only one vector. Ramaswamy *et al.* [11] proposed a method, in which n largest kNN distances are considered as outliers. This can be seen as "sparseness estimate" of a vector, in which the n sparsest vectors are considered as outliers. We name the method RRS according to the original authors' initials.

In this paper, we propose two density-based outlier detection methods. In the first method, a vector is defined as an outlier if it participates in at most T neighbourhoods in kNN graph, where threshold T is a control parameter. To accomplish this we consider kNN graph as a directed proximity graph, where the vectors are vertices of the graph and edges are distances between the vectors. We classify a vector as outlier on basis of its *indegree number* in the graph. The second method, a modification of RRS, sorts all vectors by their average kNN distances, for which a global threshold T is defined. Vectors with large average kNN -distance are all marked as outliers.

Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04)
1051-4651/04 $ 20.00 IEEE

IEEE COMPUTER SOCIETY

2. Outlier Detection with kNN Graph

2.1. kNN Graph

We define k-nearest neighbour (kNN) graph as a weighted directed graph, in which every vertex represents a single vector, and the edges correspond to pointers to neighbour vectors. Every vertex has exactly k edges to the k nearest vectors according to a given distance function. Weight of the edge e_{ij} is the distance between vectors v_i and v_j. The problem of creating kNN graph is known in computational geometry as *all-k-nearest neighbours problem* [2]. The graph can be constructed by exhaustive search considering all pairwise distances at the cost of $O(N^2)$ time. Callahan and Kosaraju [2] have shown that all-k-nearest neighbour problem can be solved in $O(kN+N\log N)$ time. The kNN graph can be used for solving clustering problem as in [3].

2.2. Detecting Outliers with kNN Graph

The kNN graph can also be used for detecting outliers [1]. Mutual k-Nearest Neighbour (MkNN) uses a special case of kNN graph. It defines an *undirected* proximity graph, which has an edge between vertices v_i and v_j if kNN graph has an edge both from v_i to v_j and from v_j to v_i. Connected components form clusters in the data and connected component with just one vertex is defined as an outlier. Potential problem with this definition is that, an outlier that is too close to an inlier, can be missclassified. For example in Fig. 1, where 13 and 16 are neighbours of each other, and not outliers according to MkNN algorithm. Thus we need more flexibility in the outlier definition.

Figure 1. Outliers in HR dataset detected with ODIN, with threshold T = 0

Ramaswamy *et al.* [11] presented *RRS method*, which calculates kNN sparseness estimate for all vectors in dataset S. Vectors are sorted in an ascending order according to the distance from a vector to its kth neighbour. Outliers are defined as the last n vectors in the ordered list. The intuitive idea is that when the distance to the kth vector is large, vector is in sparse region, and is very likely to be an outlier. A drawback of the RRS is that user has to know in advance how many outliers there are in the dataset.

2.3. Proposed Methods

We propose the following definition of outlier using kNN graph:

Definition 1 *Given kNN graph G for dataset S, outlier is a vertex, whose indegree is less than equal to T.*

First, a kNN graph is created for datset S. Then, if vertex i has an indegree of T or less, mark it as an outlier and otherwise mark it as an inlier. The proposed method has two control parameters: the number of outgoing edges k and the indegree threshold T.

Algorithm 1 ODIN

```
T is indegree threshold
Calculate kNN graph of S
for i = 1 to |S| do
  if indegree of v_i ≤ T then
    Mark v_i as outlier
  end if
end for
```

Fig. 1 shows results of ODIN with indegree threshold set to $T = 0$ applied on the dataset Hertzsprung-Russell [12] with $k = 3$ and $k = 4$. The algorithm detects star 6 correctly as an outlier with $k \geq 4$, but star 13 is not detected with any k value. However, when using threshold $T = 1$ $k = 7$ we detect stars 6 and 13 correctly as outliers.

We extend the RRS method to specify cut point in the sorted list by considering adjacent differences as shown in Fig. 2. We consider two different variants of the RRS method, mean of kNN distances (MeanDIST) and maximun of kNN distances (KDIST). When scanning the ordered list from smaller to larger distances, we check if difference between adjancent distances is larger than a given threshold. We then define vectors after the cut point as outliers. We define the threshold as:

$$T = \max(L_i - L_{i-1}) * t, \tag{1}$$

where L_i is the KDIST or MeanDIST of ith vector, and $t \in]0, 1[$ is a user defined parameter.

In Fig. 2 kNN distances are first sorted in descending order, and then the differences are taken from adjancent distances. We can see that differences grow fast when moving to right.

Figure 2. Differencies of distances for KDD dataset

Algorithm 2 MeanDIST

Compute T using Eq. 1 with t
Calculate kNN graph of S
$L \leftarrow$ Sort vectors in ascending order by kNN density
Find smallest i for which $L_i - L_{i-1} \geq T$
Mark $L_i, \ldots, L_{|S|}$ as outliers

3. Experiments

3.1. Description of Datasets

Experiments were run on *HR*, *KDD*, *NHL1*, *NHL2* and *synthetic* datasets, see Table 1. HR dataset in Fig. 1 is *Hertzsprung-Russell diagram* of the star cluster CYG OB1, where the first attribute is the logarithm effective temperature of the surface and the second the logarithm of light intensity.

Table 1. Datasets used in the experiments

Name	N	d	Outliers
HR [12]	47	2	2
KDD [9]	60318	3	486
NHL1 [8]	681	3	2
NHL2 [8]	731	3	1
synthetic	5165	2	165

KDD dataset was extracted from KDD Cup 1999 network intrusion dataset[1]. It was intended to be used as a training set for a supervised learning method. We follow the methodology described by Yaminishi *et al.* [14].

NHL1 and NHL2 dataset were selected from National Hockey League 96 player performance statistics [8]. In dataset NHL1, we selected attributes *games played*, *goals scored* and *shooting percentage*, and in NHL2 dataset *points scored*, *plus-minus statistics* and the *number of penalty minutes*. In NHL1, Chris Osgood and Mario Lemieux have been considered as outliers and Vladimir Konstantinov in NHL2 [8].

[1]Original data can be found from (http://kdd.ics.uci.edu/).

The synthetic dataset was made by generating cluster centers randomly so that they were not closer to each other than a predefined limit. Data points were then generated for each cluster with the limitation that points were not allowed to be farther than the given distance from the cluster center they belong to. We generated a GMM and using it computed minimum probability density for the points in data. Outliers were sampled from uniform distribution so that the probability density according to the GMM was at most half of the minimum of that of the data points. This ensured that the outliers were not inside the clusters.

3.2. Results

We use *Receiver Operating Characteristics* (ROC) as an evaluation method. It consists of *False Rejection* (FR) and *False Acceptance* (FA) rates. FR is number of detected outliers divided by all detections and FA is number of inliers detected as outliers divided by all detections. To combine FR and FA values we calculate *Half Total Error Rate* (HTER), defined as (FR + FA) / 2. Similar evaluation methodology has been used in [10].

Figure 3. Error rate as a function of neighbourhood size for synthetic dataset

Table 2 summarizes parameters that give minimum error rate for each algorithm. HTER is used as the error rate and the values in the parenthesis are k and T. ODIN performs well on all datasets, but for synthetic dataset MeanDIST and KDIST perform better. Good performance can be explained by the generation method of data: outliers were drawn from uniform distribution and made sure that they lie far enough from normally distributed clusters. On the other hand, results for KDD dataset show that ODIN, MeanDIST and KDIST achieve practically the same error rate. For HR, NHL1 and NHL2 datasets ODIN achieves zero error, whereas other methods perform considerably worse. Reason for large HTER values is that given a small number of outliers even a few false acceptances increases the error rate greatly.

In Fig. 3, HTER is shown as a function of neighbourhood size for synthetic dataset. MeanDIST achieves lowest error

Table 2. Summary of results as error rate (k, threshold)

Method	synthetic	KDD	HR	NHL1	NHL2
MkNN [1]	50.0 (13)	77.0 (1)	25.0 (5)	25.0 (29)	44.4 (28)
ODIN	9.0 (190,26)	49.6 (1,2)	**0.0** (7, 1)	**0.0** (87, 9)	**0.0** (36, 2)
MeanDIST	**4.9** (21, 0.05)	49.6 (232, 0.19)	30.0 (1, 0.15)	16.7 (20, 0.05)	43.8 (1, 0.57)
KDIST [11]	5.7 (12, 0.06)	**48.6** (72, 0.40)	30.0 (1, 0.15)	30.0 (1, 0.02)	41.7 (7, 0.75)

and in general has lower error rate than KDIST for synthetic data.

Figure 4. Error rate as a function of k and threshold for KDD dataset

Table 2 shows that optimal parameters for each dataset vary greatly, this leads to a problem of how to find the optimal parameter combiunation in the 2d parameter space. On the other hand, Fig. 4 shows the error rate as a function of k and T for MeanDIST with the KDD dataset. We can see that when T is below 0.1, the selection of the neighbourhood degree k is not critical. The problem of how to find the correct parameters is then just finding the best T.

4. Conclusions

We proposed a graph based outlier detection algorithm, which works well for the tested data sets. While MeanDIST and KDIST outperform the ODIN with the synthetic data, they give worse results for real data sets. This may be due to the small size of the data sets, in which case the density based methods may not obtain a reliable estimate. The proposed variant of RRS (MeanDIST) performs much better than KDIST with one dataset while the differences between the two are small in other cases.

References

[1] M. R. Brito, E. L. Chávez, A. J. Quiroz, and J. E. Yukich. Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. *Statistics & Probability Letters*, 35(1):33–42, August 1997.

[2] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the Association for Computing Machinery*, 42(1):67–90, 1995.

[3] P. Fränti, O. Virmajoki, and V. Hautamäki. Graph-based agglomerative clustering. In *Proceedings of The Third IEEE Int. Conf. on Data Mining*, pages 525–528, Melbourne, Florida, November 2003.

[4] D. Freedman, R. Purves, and R. Pisani. *Statistics*. W.W. Norton, New York, 1978.

[5] S. Guha, R. Rastogi, and K. Shim. CURE an efficient clustering algorithm for large databases. In *Proceedings of the 1998 ACM SIGMOD Int. Conf. on Management of Data*, pages 73–84, Seattle, Washington, June 1998.

[6] D. M. Hawkins. *Identification of Outliers*. Chapman and Hall, London, 1980.

[7] W. Jin, A. K. H. Tung, and J. Han. Finding top-n local outliers in large database. In *7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 293–298, 2001.

[8] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proceedings 24th Int. Conf. Very Large Data Bases*, pages 392–403, New York, USA, 1998.

[9] W. Lee, S. J. Stolfo, and K. W. Mok. A data mining framework for building intrusion detection models. In *IEEE Symposium on Security and Privacy*, pages 120–132, May 1999.

[10] J. Liu and P. Gader. Outlier rejection with mlps and variants of RBF networks. In *Proceedings of The 15th Int. Conf. on Pattern Recognition*, pages 680–683, 2000.

[11] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD Int. Conf. on Management of Data*, pages 427–438, Dallas, Texas, May 2000.

[12] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. John Wiley and Sons, New York, October 1987.

[13] G. Williams, R. Baxter, H. He, S. Hawkings, and L. Gu. A comparative study of RNN for outlier detection in data mining. In *Proceedings of the 2nd IEEE Int. Conf. on Data Mining*, Maebashi City, Japan, December 2002.

[14] K. Yamanishi, J. Takeuchi, G. Williams, and P. Milne. Online unsupervised outlier detection using finite mixtures with discounting learning algorithm. In *Proceedings The Sixth ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 320–324, August 2000.

[15] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, 1997.

3

Publication P3

V. Hautamäki, S. Cherednichenko, I. Kärkkäinen, T. Kinnunen and P. Fränti, "Improving *k*-means by outlier removal", in *Proceedings of the 14th Scandinavian Conference on Image Analysis* (*SCIA* 2005), pp. 978-987, Joensuu, Finland, June 2005.

Copyright © 2005 Springer-Verlag. Reprinted with permission.

Improving K-Means by Outlier Removal

Ville Hautamäki, Svetlana Cherednichenko, Ismo Kärkkäinen,
Tomi Kinnunen, and Pasi Fränti

Speech and Image Processing Unit,
Department of Computer Science, University of Joensuu,
P.O. Box 111, FI-80101, Joensuu, Finland
{villeh, schered, iak, tkinnu, franti}@cs.joensuu.fi

Abstract. We present an Outlier Removal Clustering (ORC) algorithm that provides outlier detection and data clustering simultaneously. The method employs both clustering and outlier discovery to improve estimation of the centroids of the generative distribution. The proposed algorithm consists of two stages. The first stage consist of purely K-means process, while the second stage iteratively removes the vectors which are far from their cluster centroids. We provide experimental results on three different synthetic datasets and three map images which were corrupted by lossy compression. The results indicate that the proposed method has a lower error on datasets with overlapping clusters than the competing methods.

1 Introduction

K-means [1] is an iterative clustering algorithm widely used in pattern recognition and data mining for finding statistical structures in data. K-means takes a training set $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and a desired number of clusters M as its input and produces a *codebook* $C = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_M\}$. The elements $\boldsymbol{c}_i$ are called *code vectors*, and they define partition of the input space into disjoint sets $\{P_1, \ldots, P_M\}$ so that $P_i = \{\boldsymbol{x} \in \mathbb{R}^d; \|\boldsymbol{x} - \boldsymbol{c}_i\| \leq \|\boldsymbol{x} - \boldsymbol{c}_j\| \ \ \forall j \neq i\}$. These sets are called *clusters*. As the clusters are disjoint, each input vector $\boldsymbol{x}_i \in X$ belongs to exactly one cluster, the one whose code vector is nearest to $\boldsymbol{x}_i$. Cluster index is denoted here as $p_i \in \{1, 2, \ldots, M\}$. The *size* of a cluster is defined as the number of input vectors assigned to the cluster and will be denoted here as n_i.

K-means starts with an initial codebook C and partition P it and improves them iteratively. If K-means has been initialized well, the resulting code vectors tend to be located at locally dense regions of the input space. In this way, K-means can be considered as a nonparametric *density estimator* which attempts to fit a model C into the input data. Text-independent speaker recognition is an example in which K-means is used in the role of a density estimator [2]. There each cluster can be thought as roughly representing a single phonetic class, and each codebook representing the distribution of the speaker's characteristic vocal space.

Given the density estimation property of K-means, it is desirable that the code vectors would be located at the correct locations, or in other words, at

H. Kalviainen et al. (Eds.): SCIA 2005, LNCS 3540, pp. 978–987, 2005.
© Springer-Verlag Berlin Heidelberg 2005

the centres of the actual clusters that generated X. However, even if the initial code vectors would be located *exactly* at the true locations, there is no guarantee that these would be the final estimated centroids. This happens more easily for overlapping clusters, see Fig. 1 for an illustration. In this figure, two Gaussian clusters having same variance have been generated by drawing $N = 200$ independent random samples from each class, and the distance between the mean vectors is varied in the three panels. The triangles denote the true centroids of the clusters and the open circles denote the estimated centroids using K-means. The true mean vectors have been used as the initial codebook to K-means. The dotted line shows the Voronoi partitioning based on the empirical centroids.

Fig. 1. Problem of K-means as a density estimator

In this trivial case, we can observe that for an increased cluster overlap, the estimated cluster centroids deviate more from the true ones. The reason for this is a fundamental problem of K-means: the clusters are disjoint, and the training vectors assigned to a wrong cluster deviate the estimated centroids away from the true ones. Figure 2 shows the error between the estimated codebook and the true codebook as a function of the distance between the cluster centers $\|\boldsymbol{c}_1 - \boldsymbol{c}_2\|$ and the ratio of their standard deviations σ_1/σ_2. Based on these observations, the usefulness of K-means as a density estimator is questionable.

Fig. 2. Error between the true and estimated models

Parametric and fuzzy clustering models such as *Gaussian Mixture Models* (GMM) algorithm [3] and *Fuzzy C-means* (FCM) [4] can be used to attack the problem of nonoverlapping clusters. However, K-means remains probably the most widely used clustering method, because it is simple to implement and provides reasonably good results in most cases. In this paper, we improve the K-means based density estimation by embedding a simple *outlier removal* procedure in the algorithm. The proposed method iteratively removes vectors far from the currently estimated codevectors. By modifying the training set X in this way, we compensate for the nonoverlapping cluster assumption. However, the algorithm should remove also points that are in the overlapping regions of clusters.

2 Outlier Removal Followed by Clustering

Outlier is defined as a noisy observation, which does not fit to the assumed model that generated the data. In clustering, outliers are considered as observations that should be removed in order to make clustering more reliable [5].

In outlier detection methods based on clustering, outlier is defined to be an observation that does not fit to the overall clustering pattern [6]. The ability to detect outliers can be improved using a combined perspective of outlier detection and clustering. Some clustering algorithms, for example DBSCAN [7] and ROCK [8], handle outliers as special observations, but their main concern is clustering the dataset, not detecting outliers.

Outlier Detection using Indegree Number (ODIN) [9] is a local density-based outlier detection algorithm. Local density based scheme can be used in cluster thinning. Outlier removal algorithm can remove vectors from the overlapping regions between clusters, if the assumption holds that the regions are of relatively low density. Higher density is found near the cluster centroid. An obvious approach to use outlier rejection in the cluster thinning is as follows: (i) eliminate outliers (ii) cluster the data using any method.

Algorithm 1. ODIN+K-means(k, T)

$\{\mathrm{ind}(\boldsymbol{x}_i) | i = 1, \ldots, N\} \leftarrow$ Calculate kNN graph
for $i \leftarrow 1, \ldots, N$ **do**
  $o_i \leftarrow 1 \big/ (\mathrm{ind}(\boldsymbol{x}_i) + 1)$
  if $o_i > T$ **then**
    $X \leftarrow X \setminus \{\boldsymbol{x}_i\}$
  end if
end for
$(C, P) \leftarrow$ K-means(X)

In this paper, we compare the proposed method against aforementioned scheme, in which the outlier removal method is ODIN and the clustering algorithm K-means. In ODIN, the outliers are defined using a k-nearest neighbour

(kNN) graph, in which every vertex represents a data vector, and the edges are pointers to neighbouring k vectors. The weight of an edge e_{ij} is $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$.

In ODIN, the outlyingess of $\boldsymbol{x}_i$ is defined as:

$$o_i = \frac{1}{\mathrm{ind}(\boldsymbol{x}_i) + 1} \ , \tag{1}$$

where $\mathrm{ind}(\boldsymbol{x}_i)$ is the *indegree* of the vertex $\boldsymbol{x}_i$, i.e. the number of edges pointing to $\boldsymbol{x}_i$. In the first step of ODIN, a kNN graph is created for the dataset X. Then, each vertex is visited to check if its outlyingness is above threshold T. Fig. 3 shows an example of kNN graph and the outlyingness values calculated for three vectors.

Fig. 3. Example of outlyingness factors in ODIN

3 Proposed Method

The objective of the proposed algorithm that we call *outlier removal clustering* (ORC), is to produce a codebook as close as possible to the mean vector parameters that generated the original data. It consists of two consecutive stages, which are repeated several times. In the first stage, we perform K-means algorithm until convergence, and in the second stage, we assign an outlyingness factor for each vector. Factor depends on its distance from the cluster centroid. Then algorithm iterations start, with first finding the vector with maximum distance to the partition centroid $d_{\max}$:

$$d_{\max} = \max_i \{\|\boldsymbol{x}_i - \boldsymbol{c}_{p_i}\|\}, \quad i = 1, \ldots, N \ . \tag{2}$$

Outlyingness factors for each vector are then calculated. We define the outlyingness of a vector $\boldsymbol{x}_i$ as follows:

$$o_i = \frac{\|\boldsymbol{x}_i - \boldsymbol{c}_{p_i}\|}{d_{\max}} \ . \tag{3}$$

We see that all outlyingness factors of the dataset are normalized to the scale $[0, 1]$. The greater the value, the more likely the vector is an outlier. An example of dataset clustered in three clusters and calculated outlyingness factors is shown in Fig. 4.

Fig. 4. Example of outlyingness factors in ORC

Algorithm 2. ORC(I, T)

```
C ← Run K-means with multiple initial solutions, pick best C
for j ← 1, ..., I do
  d_max ← max_i{||x_i − c_{p_i}||}
  for i ← 1, ..., N do
    o_i = ||x_i − c_{p_i}|| / d_max
    if o_i > T then
      X ← X \ {x_i}
    end if
  end for
  (C, P) ← K-means(X, C)
end for
```

The vectors for which $o_i > T$, are defined as outliers and removed from the dataset. At the end of each iteration, K-means is run with previous the C as the initial codebook, so new solution will be a fine-tuned solution for the reduced dataset. By setting the threshold to $T < 1$, at least one vector is removed. Thus, increasing the number of iterations and decreasing the threshold will in effect remove more vectors from the dataset, possibly all vectors.

Fig. 5 shows an example of running the proposed method on a dataset with strongly overlapping cluster so that even the cluster boundaries are not easily observable. The black dots are the original centroids. We see that with 40 iterations clusters are little bit separated and with 70 iterations clusters are totally separated.

4 Experiments

We run experiments on three synthetic datasets denoted as A1, S3 and S4 [10], which are shown in Fig. 6 and summarized in Table 1. The original cluster

Fig. 5. Example of ORC. Original dataset (left), after 40 iterations (center), and after 70 iterations (right). The removal threshold is set to $T = 0.95$ in all cases

centroids are also shown in the same figure. Vectors in datasets are drawn from multinormal distributions. In dataset A1, the clusters are fairly well separated. In dataset S3, the clusters are slightly overlapping, and in dataset S4, the clusters are highly overlapping.

Fig. 6. Datasets, A1 (left), S3 (center) and S4 (right)

We run experiments also on three map image datasets (M1, M2 and M3), which are shown in Fig. 7. Map images are distorted by compressing them with a JPEG lossy compression method. The objective is to use color quantization for finding as close approximation of the original colors as possible. JPEG compression of map images creates so-called *ringing* around the edges due to the quantization of the cosine function coefficients. In [11], color quantization methods were used to find the original colors. We apply the proposed algorithm to this problem, and we assume that the number of colors is known in advance.

We calculate *mean absolute error* (MAE) to measure the difference between the empirical codebook and the generative codebook. For ODIN with K-means, we vary both the neighbourhood size k and the number of vectors removed. For ORC, we vary the number of iterations I and the threshold T.

Table 1. Summary of the datasets

Dataset	N	M	Dataset	N	M
A1	3000	20	M1	73714	5
S3	5000	15	M2	126246	6
S4	5000	15	M3	69115	5

Fig. 7. Sample 256x256 pixel fragment from the test images M1 (left), M2 (center) and M3 (right)

Fig. 8. Results for the dataset A1 for ORC (left) and for ODIN with K-means (right)

4.1 Results

Fig. 8 shows the results for the dataset A1. We observe that increasing the parameters in the algorithms increases the error. Fig. 9 shows the results for the dataset S3. Situation without ORC iterations and threshold is shown in the back corner (in contrast to the previous figure, due to the shape of the error surface). ODIN has two "valleys", where distortion values are lower, but the error is consistently decreasing as iterations proceed or the threshold is decreased.

Fig. 10 shows the results for the dataset S4. Again, ODIN with K-means has two "valleys" where the error is lower. Regarding the number of remaining vectors, we see that the more vectors we remove with the ORC algorithm, the

Fig. 9. Results for the dataset S3 for ORC (left) and for ODIN with K-means (right)

better the accuracy will be. This is because the ORC algorithm works as designed for S4 dataset by removing vectors that reside between clusters. On the other hand, when increasing parameters in ODIN algorithm first, we get lower error and then the error starts to increase.

Results for the M1 - M3 datasets running the ORC algorithm are presented in Figs. 11 and 12. We note that for all the test cases, ORC reaches lower error when the number of iterations is increased enough or the threshold is decreased. Error surface of the dataset M1 has an interesting behaviour, where error first increases and then it starts to decrease. Error surfaces for ODIN are omitted as with all parameter combinations the error increases in relation to the standard K-means.

Fig. 10. Results for the dataset S4 for ORC (left) and for ODIN K-means (right)

Fig. 11. Results for the datasets M1 (left) and M2 (right) for ORC

Fig. 12. Results for the dataset M3 for ORC

Table 2. Best MAEs obtained

Algorithm	A1	S3	S4	M1	M2	M3
Plain K-means	60	5719	7100	47	32	26
ODIN + K-means	58	4439	4754	61	48	45
ORC	56	3329	2813	45	13	23

In Table 2, we show the smallest MAE between original codebook and those obtained by using K-means, ODIN with K-means and ORC. The results indicate potential of the proposed method. The ORC algorithm outperforms the traditional K-means and K-means preceded by outlier removal for all three data sets. For the non-overlapping data set (A1), the results are close to each other. However, when cluster overlap is increased, the proposed algorithm shows substantially improved performance over the baseline methods. For the most difficult data set (S4), the proposed method gives 1.5 - 2 times smaller error. Although parameter setting might be a difficult depending on the dataset. For the map image datasets, ORC performs systematically better than K-means in all cases. With datasets M1 and M3, ORC and K-means are close to each other in performance, but for M2 ORC more than halves the error in relation to K-means.

5 Conclusions

In this paper, we have proposed to integrate outlier removal into K-means clustering (ORC) for nonparametric model estimation. The proposed method was also compared with the standard K-means without outlier removal, and a simple approach in which outlier removal precedes the actual clustering. The proposed method was evaluated on three synthetic data sets with known parameters of the generative distribution and three map image datasets with known cluster centroids.

The test results show that the method outperforms the two baseline methods, particularly in the case of heavily overlapping clusters. A drawback is that correct parameter setting seems to depend on the dataset. Thus, the parameter setting should be automatized in future.

References

1. Linde, Y., Buzo, A., Gray, R.M.: An algorithm for vector quantizer desing. IEEE Transactions on Communications **28** (1980) 84–95
2. Kinnunen, T., Karpov, E., Fränti, P.: Real-time speaker identification and verification. IEEE Transactions on Speech and Audio Processing (2005) Accepted for publication.
3. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society B **39** (1977) 1–38

4. Dunn, J.: A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. Journal of Cybernetics **3** (1974) 32–57
5. Guha, S., Rastogi, R., Shim, K.: CURE an efficient clustering algorithm for large databases. In: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, Seattle, Washington (1998) 73–84
6. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: A new data clustering algorithm and its applications. Data Mining and Knowledge Discovery **1** (1997) 141–182
7. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: 2nd International Conference on Knowledge Discovery and Data Mining. (1996) 226–231
8. Guha, S., Rastogi, R., Shim, K.: ROCK: A robust clustering algorithm for categorical attributes. In: 15th International Conference on Data Engineering. (1999) 512–521
9. Hautamäki, V., Kärkkäinen, I., Fränti, P.: Outlier detection using k-nearest neighbour graph. In: 17th International Conference on Pattern Recognition (ICPR 2004), Cambridge, United Kingdom (2004) 430–433
10. Virmajoki, O.: Pairwise Nearest Neighbor Method Revisited. PhD thesis, University of Joensuu, Joensuu, Finland (2004)
11. Kopylov, P., Fränti, P.: Color quantization of map images. In: IASTED Conference on Visualization, Imaging, and Image Processing (VIIP'04), Marbella, Spain (2004) 837–842

4

Publication P4

V. Hautamäki, T. Kinnunen, I. Kärkkäinen, J. Saastamoinen, M. Tuononen and P. Fränti, "Maximum *a posteriori* adaptation of the centroid model for speaker verification", *IEEE Signal Processing Letters*, Vol. 15, pp. 162-165, 2008.

Copyright © 2008 IEEE. Reprinted with permission.

Maximum *a Posteriori* Adaptation of the Centroid Model for Speaker Verification

Ville Hautamäki, Tomi Kinnunen, Ismo Kärkkäinen, Juhani Saastamoinen, Marko Tuononen, and Pasi Fränti

***Abstract*—Maximum a posteriori adapted Gaussian mixture model (GMM-MAP) is widely used in speaker verification. GMMs have three sets of parameters to be adapted: means, covariances, and weights. However, practice has shown that it is sufficient to adapt the means only. Motivated by this, we formulate maximum a posteriori vector quantization (VQ-MAP) procedure which stores and adapts the mean vectors (centroids) only. Experiments on the NIST 2001 and NIST 2006 corpora indicate that VQ-MAP gives comparable accuracy with GMM-MAP with simpler implementation and faster adaptation.**

***Index Terms*—Bayesian methods, centroid model, maximum a posteriori (MAP) adaptation, speaker verification, vector quantization.**

I. Introduction

In *speaker verification* [1], the unknown speech utterance is introduced to the system accompanied by a claim. The task is to decide whether the claim was true or false, by matching the unknown test utterance to a previously stored model. Speaker recognition systems typically have used generative models such as *vector quantization* (VQ) [1] (aka the *centroid model*) or *Gaussian mixture models* (GMMs) [2].

The generative model is typically trained using the maximum likelihood (ML) principle. The ML approach usually does not generalize well to unseen speech data with finite amount of training material. *Maximum a posteriori* (MAP) objective training attacks this problem of by using a so-called *universal background model* (UBM) [2]. In the MAP approach, prior knowledge of the distribution of the model parameters is incorporated into the modeling process. Even if some areas of the feature space are less represented in the training data, the prior information about the parameters can help to overcome the problem. However, incorporating the prior information is not trivial because prior parameter distribution has its own parameters, known as *hyperparameters*, which can be difficult to estimate.

Maximum *a posteriori* training for Gaussian mixtures was first formulated in [3], where Gauvain and Lee solved two key issues in MAP estimation of Gaussian mixture parameters, namely, choice of the prior distribution family, and specification of the parameters of the prior densities, which led to simple *expectation maximization* (EM) re-estimation formulae. The original motivation of the authors was to enhance the performance of a HMM-based speech recognition system. The idea was later successfully applied to speaker verification as well [2]. To overcome the parametrization problem, Reynolds *et al.* [2] proposed to use the universal background model as a set of parameters of the prior distribution. Some constraints to the parameters and a relevance factor were introduced to solve the prior parametrization problem.

Gaussian mixtures have three sets of parameters to be adapted: mean vectors (*centroids*), covariance matrices, and weights. However, experiments have indicated that best results are obtained by adapting the mean vectors only [2]. Motivated by this, we formulate maximum *a posteriori* vector quantization (VQ-MAP), which is a special case of the GMM-MAP method [2]. The advantages of the proposed centroid-based model are simpler implementation and much faster adaptation. The speedup originates mostly from the replacement of the Gaussian density computations with squared distance computations, leaving out the exponentiation and additional multiplications.

II. VQ-MAP

In vector quantization, the goal is to estimate parameter vector modeling the speaker, denoted as $\Theta = (\boldsymbol{c}_1^t, \ldots, \boldsymbol{c}_K^t)^t$. Here $\boldsymbol{c}_i$ are the centroids and K is the model size, which is a trade-off between the representation accuracy and the speaker model size.

The maximum *a posteriori* modeling paradigm, irrespective of the actual model in question, is formulated as a way to find Θ that maximizes the posterior probability density function (pdf). Formally

$$\begin{aligned}\Theta_{\mathrm{MAP}} &= \arg\max_{\Theta} p(\Theta|X) \\ &= \arg\max_{\Theta} p(X|\Theta) g(\Theta)\end{aligned} \tag{1}$$

where $p(X|\Theta)$ is the likelihood of the training set $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ given parameters Θ, and $g(\Theta)$ is the prior pdf of the parameters.

Three subproblems need to be solved so that a maximization algorithm can be derived:

- likelihood function $p(X|\Theta)$ needs to be defined in terms of vector quantization;
- appropriate prior distribution $g(\Theta)$ needs to be defined;
- prior distribution contains its own set of parameters, which also needs to be estimated.

In the following, we address these points and derive the maximization algorithm.

Manuscript received June 18, 2007; revised November 3, 2007. The work was supported by the National Technology Agency of Finland (TEKES) as the projects New Methods and Applications of Speech Technology (PUMS). The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Brian Kan-Wing Mak.

The authors are with the Speech and Image Processing Unit, Department of Computer Science and Statistics, University of Joensuu, FIN-80101 Joensuu, Finland (e-mail: villeh@cs.joensuu.fi; tkinnu@cs.joensuu.fi; iak@cs.joensuu.fi; juhani@cs.joensuu.fi; mtuonon@cs.joensuu.fi; franti@cs.joensuu.fi).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LSP.2007.914792

1070-9908/$25.00 © 2008 IEEE

A. Modeling VQ as a Gaussian Mixture

For the VQ-MAP algorithm, we must formulate (1) in the vector quantization framework. Since VQ is not a parametric probabilistic model, we need to specify a likelihood pdf that corresponds to the *mean squared error* (MSE), which defines the VQ model. We can model the likelihood $p(X|\Theta)$ as a Gaussian mixture as in [4]. The density of the kth component is defined as

$$p(\boldsymbol{x}|\boldsymbol{c}_k, \Sigma_k) = \frac{1}{(2\pi\epsilon)^{1/2}} \exp\left\{-\frac{1}{2\epsilon}\|\boldsymbol{x} - \boldsymbol{c}_k\|^2\right\} \quad (2)$$

where $\Sigma_k = I\epsilon$, and ϵ is constant. It is shown in [4] that with this model, the EM algorithm reduces to k-means algorithm and that component prior weights π_k do not play any role in the algorithm. The weights just reflect the proportion of the data vectors in a given cluster.

B. Defining the Prior Density

When selecting the appropriate prior density $g(\Theta)$, a good choice would be the *conjugate prior* of the $p(\boldsymbol{x}|\boldsymbol{c}_k, \Sigma_k)$ as in [4]. Prior distribution is called a conjugate prior, if its algebraic form is the same as the resulting posterior distribution. The conjugate prior of a multivariate Gaussian with a known covariance matrix is a multivariate Gaussian. We can therefore model the prior of the component k as

$$g(\boldsymbol{c}_k|\boldsymbol{\mu}_k, \hat{\Sigma}_k) = B_k \exp\left\{-\frac{1}{2}(\boldsymbol{c}_k - \boldsymbol{\mu}_k)^t \hat{\Sigma}_k^{-1}(\boldsymbol{c}_k - \boldsymbol{\mu}_k)\right\} \quad (3)$$

where $\hat{\Sigma}$ is the covariance matrix of the prior distribution, and

$$B_k = \frac{1}{(2\pi)^{D/2}|\hat{\Sigma}_k|^{1/2}}. \quad (4)$$

Assuming independence between the parameters of the individual mixture components, as was done in [3], we conclude the definition of the prior model by noting that

$$g(\Theta) \propto \prod_{k=1}^{K} g(\boldsymbol{c}_k|\boldsymbol{\mu}_k, \hat{\Sigma}_k). \quad (5)$$

C. MAP Estimates for Vector Quantization

In order to maximize the posterior pdf in (1), we need to jointly determine the observation posteriors and the model parameters of each component. Unfortunately, the maximization cannot be performed directly [3]. Instead, locally optimal solution can be obtained by EM algorithm [4] for the Gaussian mixture models and by k-means algorithm for the vector quantization models. Both algorithms work essentially in a similar manner:

1) find observation posteriors (E-step);
2) given the posteriors, re-estimate the parameters (M-step).

In k-means, the observation posteriors correspond to hard partitioning of the dataset. In the M-step, the parameters are maximized by calculating new centroid estimates. Now the corresponding steps need to be defined in the new framework so that MAP parameters can be optimized.

Interestingly, the term $g(\Theta)$ affects the maximization of the posterior distribution only in the M-step [4]. Optimal Θ with respect to observation posteriors can then be calculated by maximizing the following auxiliary function [3]:

$$R(\Theta, \hat{\Theta}) = Q(\Theta, \hat{\Theta}) + \log g(\Theta) \quad (6)$$

where $\hat{\Theta}$ are the parameters estimated in the previous iteration, and Θ are the parameters to be estimated. The function Q is the expectation of the complete-data log likelihood and can be expressed as [4]

$$Q(\Theta, \hat{\Theta}) = \sum_{i=1}^{N}\sum_{j=k}^{K} \gamma_{ik}\left\{\ln \pi_k + \ln p(\boldsymbol{x}|\boldsymbol{c}_k, \Sigma_k)\right\} \quad (7)$$

where π_k is the prior weight of the component k, and γ_{ik} is the posterior probability of the observation i for the component k. By letting $\epsilon \rightarrow 0$ in (2), the complete-data log likelihood function becomes the MSE [4]

$$Q(\Theta, \hat{\Theta}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik}\|\boldsymbol{x}_i - \boldsymbol{c}_k\|^2. \quad (8)$$

The values of r_{ik} form a binary matrix, where

$$r_{ik} = \begin{cases} 1, & \text{if } k = \arg\min_j \|\boldsymbol{x}_i - \boldsymbol{c}_j\| \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

In vector quantization literature, MSE describes the distortion when observations $\boldsymbol{x}_i$ are encoded as their nearest centroids $\boldsymbol{c}_j$.

By substituting (5) and (8) into (6), we arrive at a new auxiliary function form

$$R(\Theta, \hat{\Theta}) = -\sum_{i=1}^{N}\sum_{j=1}^{K} r_{ik}\|\boldsymbol{x}_i - \boldsymbol{c}_k\|^2 - \sum_{k=1}^{K}(\boldsymbol{c}_k - \boldsymbol{\mu}_k)^t \hat{\Sigma}_k^{-1}(\boldsymbol{c}_k - \boldsymbol{\mu}_k). \quad (10)$$

We need to find such $\boldsymbol{c}_k$ for each component that minimizes the above equation. The $\boldsymbol{\mu}_k$ and $\hat{\Sigma}_k$ are our prior parameters for the component k, and they are selected from a previously trained universal background model as in [2]. However, in the VQ model, covariance matrices (variance parameters) are not recorded as a part of the UBM. Therefore, we substitute $\hat{\Sigma}_k = I$ for all components. This is motivated by our model assumptions in (2). Now, $R(\Theta, \hat{\Theta})$ can be written as

$$R(\Theta, \hat{\Theta}) = r_{11}\|\boldsymbol{x}_1 - \boldsymbol{c}_1\|^2 + \cdots + r_{NK}\|\boldsymbol{x}_N - \boldsymbol{c}_K\|^2 + \|\boldsymbol{c}_1 - \boldsymbol{\mu}_1\|^2 + \cdots + \|\boldsymbol{c}_K - \boldsymbol{\mu}_K\|^2. \quad (11)$$

Now, let us denote by $S_k = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{|S_k|}\}$ the set of training vectors that are mapped to $\boldsymbol{c}_k$. We denote by R_k the terms of $R(\Theta, \hat{\Theta})$ that contain centroid $\boldsymbol{c}_k$

$$\begin{aligned} R_k &= \|\boldsymbol{x}_1 - \boldsymbol{c}_k\|^2 + \cdots + \|\boldsymbol{x}_{|S_k|} - \boldsymbol{c}_k\|^2 + \|\boldsymbol{c}_k - \boldsymbol{\mu}_k\|^2 \\ &= 2|S_k|\langle \bar{\boldsymbol{x}}_k, \boldsymbol{c}_k\rangle + |S_k|\|\boldsymbol{c}_k\|^2 + \|\boldsymbol{c}_k - \boldsymbol{\mu}_k\|^2 \\ &= 2|S_k|\langle \bar{\boldsymbol{x}}_k, \boldsymbol{c}_k\rangle + (|S_k| + 1)\|\boldsymbol{c}_k\|^2 + 2\langle \boldsymbol{c}_k, \boldsymbol{\mu}_k\rangle \end{aligned} \quad (12)$$

where $|S_k|$ is the number of vectors mapped to centroid $\boldsymbol{c}_k$, and $\bar{\boldsymbol{x}}_k$ is the average of all vectors in the same cluster. Taking the gradient with respect to $\boldsymbol{c}_k$ from (12), we obtain the centroid re-estimation formula for the M-step as

$$\boldsymbol{c}_k = \frac{|S_k|}{|S_k|+1}\bar{\boldsymbol{x}}_k + \frac{1}{|S_k|+1}\boldsymbol{\mu}_k. \tag{13}$$

III. Speaker Verification Using VQ-MAP

In speaker verification, the prior distribution of the speaker model parameters is represented by a universal background model, which is created by collecting a large number of representative training utterances from a number of speakers. These utterances are then converted into sequences of spectral feature vectors and pooled into a single training set. The UBM can be trained using any clustering algorithm such as k-means, and it is presented by a set of centroid vectors denoted as $U = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K\}$.

A. Enrollment Phase

A new speaker is enrolled by adapting the UBM with the MAP procedure. Given the training data for a new speaker, $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$, the adaptation is performed using the following steps:

1) initialization: Set $\boldsymbol{c}_k = \boldsymbol{\mu}_k$ for $k = 1, 2, \ldots, K$;
2) for $i = 1, 2, \ldots, I$, do
   a) For each training vector $\boldsymbol{x}_n$, find the index of the nearest neighbor in the adapted model
   $$q_n = \arg\min_{1 \le k \le K} \|\boldsymbol{x}_n - \boldsymbol{c}_k\|^2. \tag{14}$$
   b) For the kth cluster, define the set of vectors mapped into that cluster as $S_k = \{\boldsymbol{x}_n \in X | q_n = k\}$ and calculate the centroid vector
   $$\bar{\boldsymbol{x}}_k = \frac{1}{|S_k|}\sum_{\boldsymbol{x}_n \in S_k} \boldsymbol{x}_n. \tag{15}$$
   c) For the kth cluster, update the adapted vector as
   $$\boldsymbol{c}_k = w_k \bar{\boldsymbol{x}}_k + (1 - w_k)\boldsymbol{\mu}_k \tag{16}$$
   where
   $$w_k = \frac{|S_k|}{|S_k| + r} \tag{17}$$
   and r is a fixed constant known as the *relevance factor*.

The result is the adapted model $C = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_K\}$. The adaptation process is demonstrated in Fig. 1.

The relevance factor r is a parameter that is used to control the balance of how much the UBM vector and the speaker-specific training data are weighted in adaptation. The "unit" of the parameter is vector count and can be given an interpretation as follows. By having $w_k > 1/2$ in (17), we have $|S_k| > r$. Thus, the relevance parameter can be interpreted as the number of training vectors that are needed in order the computed centroid $\bar{\boldsymbol{x}}_k$ to be equally reliable to the corresponding UBM vector.

B. Verification Phase

Given a sequence of feature vectors, $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T\}$, and the claimed speaker model, $C = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_K\}$, we compute the log likelihood ratio and compare it against a verification threshold to make the decision. In the VQ model, the log-likelihood is related to the negative square error given in (8). Thus, the match score can be defined as

Fig. 1. Illustration of the VQ-MAP procedure with two-dimensional data. The centroid vectors of the universal background model (•) define the prior distribution of the model parameters. Given the training data (.), the adapted model (+) is derived by making local changes to the prior model. Here, the relevance factor is 10.

$$\begin{aligned}\text{score} &= -\,\text{MSE}(X, C) - (-\text{MSE}(X, U)) \\ &= \text{MSE}(X, U) - \text{MSE}(X, C)\end{aligned} \tag{18}$$

where

$$\text{MSE}(X, Y) = \frac{1}{|X|}\sum_{\boldsymbol{x}_i \in X} \min_{\boldsymbol{y}_k \in Y} \|\boldsymbol{x}_i - \boldsymbol{y}_k\|^2. \tag{19}$$

C. Relation to GMM-MAP

In principle, the resulting adaptation (14)–(17) are simply a special case of the more general GMM-MAP equations [2]. By assuming a diagonal covariance with all dimensions sharing the same variance and by replacing the observation posteriors with the binary 0/1 values (1 for the most probable cluster and 0 for all the others) and interpreting the proportion of data vectors in each cluster as the mixture weight, GMM-MAP equals VQ-MAP. In practice, the proposed model is much simpler to implement and results in faster adaptation.

The VQ-MAP and GMM-MAP algorithms require $O(NKI)$ squared distance and Gaussian density computations, where I is the number of iterations. By counting the number of elementary operations in the squared Euclidean distance and the diagonal covariance Gaussian, the speedup ratio can be written as

$$\text{speedup} = \frac{T_{\text{exp}} + (2D+2)T_{\text{mul}} + 2DT_{\text{add}}}{DT_{\text{mul}} + 2DT_{\text{add}}} \tag{20}$$

where T_{exp}, T_{mul}, and T_{add} denote the costs of exponentiation, multiplication, and addition, respectively, and D denotes the feature vector dimensionality. Exponentiation takes more time than multiplication and addition, which yields a significant speedup in practice, as will be demonstrated.

IV. Experiments

We have used the NIST 2001 SRE corpus for optimizing the control parameters and NIST 2006 SRE corpus for validating the results.[1] The results are presented on the one-speaker detection set of each corpus. The NIST 2001 corpus contains 2 min of training material per each of the 174 target speakers and 2038 test segments. The NIST 2006 corpus contains 816 target speakers trained using 5 min of data and 3735 test segments. We have trained the UBMs using the development set of the NIST 2001 SRE corpus.

[1]http://nist.gov/speech/tests/spk/

Fig. 2. Effect of control parameters of VQ-MAP on the equal error rates of the NIST 2001 corpus using a model of size $K = 64$.

We use the first 12 mel-frequency cepstral coefficients (MFCCs), their deltas, and double deltas as the features, implying 36-dimensional feature space. A periodicity-based voice activity detector was used [5], and the detected speech vectors were normalized to have zero mean and unit variance. The UBM for VQ was generated by a recursive split algorithm, followed by fine-tuning with k-means. This model was used as the initial guess for the GMM model with diagonal covariances and fine-tuned by EM.

Experiment with different settings of the relevance parameter (r) and the number of iterations (I) indicates that, independent on the number of iterations used, the relevance factor should be set to 8 or higher as shown in Fig. 2. There is no significant difference between different iterations, which is similar behavior that was observed also for the GMM-MAP method [2]. It can also be seen in Fig. 1 that the adapted model does not change much with further iterations. We fix the parameters as $r = 12$ and $I = 2$ for further experiments.

For the GMM-MAP, we use the well-established values of $r = 16$ and $I = 1$ [2]. In addition to the means-only adapted GMM, we also present the results for GMM in which the variances are also adapted using the same relevance factor.

The detection error trade-off (DET) curves for the NIST 2001 and NIST 2006 corpuses are shown in Fig. 3. The running times on the NIST 2001 corpus have been summarized in Table I. Finally, Table II summarizes the speedup factors for different model sizes relative to two VQ-MAP iterations. The running times include only the work done in adapting the models; the overhead due to feature extraction and file I/O has been excluded.

VQ-MAP and GMM-MAP provide accuracies close to each other. The variance-adapted GMM gives slightly better result at low false acceptance rates on the NIST 2001 corpus. However, the difference vanishes on the NIST 2006 corpus which includes more training data and more difficult channel conditions.

The VQ-based systems run much faster. In particular, the adaptation step is significantly faster, even though VQ-MAP performs two iterations. An advantage of the obtained speedup would be on platforms with significantly limited CPU power and fixed-point arithmetics, such as mobile phones [6].

Fig. 3. Results for the NIST 2001 corpus (left) and NIST 2006 corpus (right), model size $K = 64$.

TABLE I
COMPARISON OF THE CPU TIME (IN SECONDS) ON THE NIST 2001 CORPUS (MODEL SIZE $K = 64$)

Method	UBM train	Adaptation avg.	Adaptation std	Matching avg.	Matching std
VQ-MAP ($I = 1$)	170	0.01	0.005	0.14	0.06
VQ-MAP ($I = 2$)	170	0.03	0.009	0.14	0.06
GMM-MAP	514	0.18	0.054	0.54	0.23
GMM-MAP +var	514	0.18	0.055	0.54	0.23

TABLE II
ADAPTATION SPEEDUP FACTORS ON THE NIST 2001 CORPUS RELATIVE TO TWO VQ-MAP ITERATIONS

Method	Model size (K) 64	128	256	512	1024
VQ-MAP ($I = 1$)	2:1	2:1	2:1	2:1	2:1
VQ-MAP ($I = 2$)	1:1	1:1	1:1	1:1	1:1
GMM-MAP	6:1	6:1	9:1	19:1	19:1
GMM-MAP +var	6:1	6:1	9:1	19:1	20:1

V. CONCLUSIONS

We have formulated the MAP algorithm originally developed for GMM to work with the VQ-based model. Experimental results show that the proposed method provides similar recognition accuracy than the GMM-based algorithm but with simpler and faster implementation.

REFERENCES

[1] F. Bimbot, J.-F. Bonastre, C. Fredouille, G. Gravier, S. Meignier, T. Merlin, J. Ortega-Garcia, I. Magrin-Chagnolleau, D. Petrovska-Delacretaz, and D. A. Reynolds, "A tutorial on text-independent speaker verification," *EURASIP J. Appl. Signal Process.*, no. 4, pp. 430–451, 2004.

[2] D. Reynolds, T. Quatieri, and R. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digit. Signal Process.*, vol. 10, no. 1, pp. 19–41, 2000.

[3] J. L. Gauvain and C.-H. Lee, "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains," *IEEE Trans. Speech Audio Process.*, vol. 2, no. 2, pp. 291–298, Apr. 1994.

[4] C. M. Bishop, *Pattern Recogntion and Machine Learning*. New York: Springer-Verlag, 2006.

[5] V. Hautamäki, M. Tuononen, T. Niemi-Laitinen, and P. Fränti, "Improving speaker verification by periodicity based voice activity detection," in *Proc. 12th Int. Conf. Speech and Computer (SPECOM'2007)*, Moscow, Russia, Oct. 2007, vol. 2, pp. 645–650.

[6] J. Saastamoinen, E. Karpov, V. Hautamäki, and P. Fränti, "Accuracy of MFCC based speaker recognition in Series 60 device," *J. Appl. Signal Process.*, no. 17, pp. 2816–2827, Sep. 2005.

5

Publication P5

V. Hautamäki, T. Kinnunen and P. Fränti, "Text-independent speaker recognition using graph matching", *Pattern Recognition Letters*, Vol. 29, No. 9, pp. 1427-1432, July 2008.

Copyright © 2008 Elsevier. Reprinted with permission.

Pattern Recognition Letters 29 (2008) 1427–1432

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Text-independent speaker recognition using graph matching

Ville Hautamäki *, Tomi Kinnunen, Pasi Fränti

Speech and Image Processing Unit, Department of Computer Science and Statistics, University of Joensuu, P.O. Box 111, FI-80101, Joensuu, Finland

ARTICLE INFO

Article history:
Received 31 October 2006
Received in revised form 19 February 2008
Available online 7 March 2008

Communicated by O. Siohan

Keywords:
Affine transformation invariance
Graph matching
Structural matching
kNN graph
Clustering
Speaker recognition

ABSTRACT

Technical mismatches between the training and matching conditions adversely affect the performance of a speaker recognition system. In this paper, we present a matching scheme which is invariant to feature rotation, translation and uniform scaling. The proposed approach uses a neighborhood graph to represent the global shape of the feature distribution. The reference and test graphs are aligned by graph matching and the match score is computed using conventional template matching. Experiments on the NIST-1999 SRE corpus indicate that the method is comparable to conventional Gaussian mixture model (GMM) and vector quantization (VQ)-based approaches.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

One of the biggest challenges in automatic speaker recognition is obtaining *invariance* across varying operating conditions, while retaining maximum speaker variability. Different handset type, transmission line/coding, and background noise are typical factors, which lead to signal mismatch across training and recognition. For a speaker recognition system to be useful in practice it needs to be optimized against the mismatch problem.

Various approaches have been proposed for tackling the invariance problem, including robust feature extraction (Mammone et al., 1996), feature normalization (Pelecanos and Sridharan, 2001), model transformation (Kenny et al., 2007; Teunen et al., 2002; Vogt and Sridharan, 2008), and match score normalization (Auckenthaler et al., 2000; Reynolds et al., 2000).

State-of-the-art text-independent speaker recognizers use mean subtraction at the utterance level, often referred to as *cepstral mean subtraction* (CMS) in the context of cepstral features. The assumption in mean subtraction is that all the feature vectors have been translated by an unknown channel-dependent vector. By subtracting the mean from both the training and testing vectors, the matching is less affected by this bias. For clean data (no channel mismatch), CMS degrades accuracy.

A general affine channel/environment model (Mak and Tsang, 2004; Mammone et al., 1996) includes rotation and scaling of the feature vectors in addition to the additive bias. The three transformations – rotation, scaling, and translation – can be collectively expressed as an affine feature distortion model: $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$. The matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ are channel-dependent transformation parameters, whereas $\boldsymbol{x}$ and $\boldsymbol{y}$ are the "clean" and the "noisy" (observed) vectors, respectively.

In image- and video-based biometrics, invariance against rotation, translation and scaling is often desirable. For instance, a face recognizer would produce the same match score, independent of face tilting (rotation), location with respect to the background (translation) and distance from the camera (scale). A natural idea to achieve invariance is to construct a graph from certain feature points from the images and then to use *graph matching* (Bunke and Shearer, 1998) methodology. In the matching phase, only the graph structures – and not the original feature points – are compared. For example, Burge and Burger (2000) use Voronoi diagram graphs to model ear shape. The graphs of the reference ear and the unknown ear were matched using error-correcting graph matching.

It is an open question whether similar ideas could be adopted to speaker recognition. In our view, formulation of a transformation invariant matching scheme for speech features poses several challenges. First, images are two-dimensional, and the semantic meaning of the constructed graph can be visually verified. However, commonly used speech spectrum features are high-dimensional (10–40 dimensions), and it is difficult to give an intuitive meaning to the graph calculated from the extracted features. Second, in text-independent speaker recognition, the feature distributions of

* Corresponding author. Fax: +358 132517955.
E-mail address: villeh@cs.joensuu.fi (V. Hautamäki).

0167-8655/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2008.02.021

Fig. 1. Speaker recognition system diagram.

the training vectors and the test utterance are likely to vary because of text mismatch in addition to the technical mismatches mentioned above. It is also unclear whether the matching should use the whole distributions, or should a good match be indicated if the *sub*-graphs from the reference vectors and the test utterance match well.

The motivation of this paper is to experiment with a few simple ideas. To our knowledge, no graph-based matching has previously been proposed for text-independent speaker recognition. The overview of the proposed scheme is illustrated in Fig. 1. We first cluster both the training and the testing vectors into a small number of clusters, represented by a set of centroid vectors. Neighborhood graphs are then constructed for both sets. Finally, structural similarity of the reference and the test graphs is evaluated by calculating the degree of isomorphism between the graphs.

We also propose a matching framework which is a hybrid between graph-based *structural matching* and vector-based *template matching*. Graph matching is used as a pairing tool between the reference and the test centroids. The paired vectors from each set are then used for finding the parameters of the affine transformation model. Finally, the match score is computed as the distortion between the compensated centroids.

Feature and speaker model transformations, including the affine transformation, have been studied by different authors (Kenny et al., 2007; Mak and Tsang, 2004; Mammone et al., 1996; Siohan and Lee, 1997; Vogt and Sridharan, 2008). These methods usually require either parallel training data recorded simultaneously through various handsets, or a large number of training utterances collected from multiple recording sessions from a number of speakers. These datasets are then used for estimating the transformation parameters. The method that we propose, in turn, aligns the test vectors to the claimed speaker's model during verification. Therefore, the proposed method does not require any external data or training of a channel/session variability model.

The rest of the paper is organized as follows. In Section 2, we give details of the structural graph matching framework. Section 3 describes the hybrid structural and template matching algorithm. Experimental setup and the results are described in Section 4. Finally, conclusions are drawn in Section 5.

2. Graph matching based on maximum common subgraph

In order to match two high-dimensional feature sets, we need to extract some stable "feature points" from each set. This is somewhat analogous to finding minutia points from fingerprint images, or eye locations from face images. We extract the same number of feature points (N) from each set. Our assumption here is that when the two feature sets originate from the same speaker, the *relative positions* of the feature points, *i.e.* the *shape of the distribution*, remains similar despite the channel effects and noise.

The steps of the proposed graph-based alignment are summarized as follows:

(1) *Feature extraction*: Extract N feature points from both the test and the training sets by clustering.
(2) *Graph creation*: Construct neighborhood graphs from the training and testing feature points.
(3) *Pairing of the feature points*: Find the maximum common subgraph (MCS) between the two graphs.
(4) *Computing the match score*: Based on the size of the maximum common subgraph.

To implement Step 1, we use k-means (MacQueen, 1967). As a result of clustering, we have a set of code vectors located in "dense" regions of the feature distribution. It is assumed that the dense regions are related to broad phonetic classes of the given speaker and that they form stable points that can be used as the reference points in alignment.

2.1. Graph creation

The output of clustering is a set of centroid vectors. We model the relationship between the centroids using a *k-nearest neighbour graph* (kNNG), in which each vertex represents a feature vector, and the edges are pointers to the neighbouring vectors (Fränti et al., 2006). Each vertex has exactly k edges to its k-nearest neighbors in the sense of Euclidean distance. The final graph is obtained by removing the distance information (weights), and converting all the edges to undirected ones as illustrated in Fig. 2. As an example, the edge set of the graph in the right side of Fig. 2 is $E = \{(a,b),(b,c),(c,d),(d,e),(g,f),(h,i),(i,j),(j,k),(l,m),(m,n)\}$. In (Fränti et al., 2006), neighbourhood size was studied in the context of agglomerative clustering. It was found that, practical values of k vary between 5 and 12. Here we set $k = 8$.

Fig. 2. Directed kNNG with $k = 1$ (left), undirected version (right).

2.2. Defining the match score

A matching function is defined between two undirected and unweighted graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where $V_{1,2}$ and $E_{1,2}$ are the discrete sets of vertices and edges, respectively. The most natural measure of similarity between two graphs is based on graph isomorphism (Diestel, 2000). Two graphs are *isomorphic* if their vertices and edges can be relabeled using a common set of labels. More precisely, the graphs G_1 and G_2 are isomorphic, if there exists a bijection from the vertices and edges of G_1 to the vertices and edges of G_2. Whenever two graphs are isomorphic, their structure (topology) is said to *match perfectly*.

Unfortunately, graphs constructed from real-world signals never match perfectly due to measurement inaccuracies and signal distortions. For that reason, we need to define a non-binary match score which measures the degree of isomorphism (Bunke and Messmer, 1997). We adopt the following distance measure (Bunke and Shearer, 1998):

$$D_{\mathrm{MCS}}(G_1, G_2) = 1 - \frac{|\mathrm{MCS}(G_1, G_2)|}{\max(|G_1|, |G_2|)}, \tag{1}$$

where $\mathrm{MCS}(G_1, G_2)$ is the *maximum common subgraph* (MCS) between G_1 and G_2, and $|\cdot|$ denotes the number of vertices in the given graph. A maximum common subgraph is defined as the largest graph, which is isomorphic to subgraphs found in G_1 and G_2. The limits of the distance function D_{MCS} are 0 and 1, where 0 is obtained when the graphs G_1 and G_2 are isomorphic. The distance reaches a maximum when the size of the MCS is 0, and this happens only when one of the graphs is empty. If both graphs contain vertices, at least the individual vertices are isomorphic to each other and D_{MCS} is less than one. In our graph matching system, $|G_1| = |G_2| = N$, so the maximum size of the MCS is N. By noting that the smallest size of MCS is 1, we obtain that (1) can have at most $N - 1$ unique values.

There are both advantages and disadvantages in using the distance (1). The advantage is that the measure (1) is theoretically a very good candidate for a similarity score: it satisfies all the metric space axioms (Bunke and Shearer, 1998), including the triangular inequality. The drawback is that finding the maximum common subgraph is an NP-complete problem (Kann, 1992). For this reason, we have to restrict the experiments to a small number of vectors.

2.3. Computing the match score

We use the *Durand–Pasari* algorithm (Durand et al., 1999) to find the maximum common subgraph. It finds the MCS in two steps. First, a so-called *association graph* (Levi, 1972) is constructed from the graphs G_1 and G_2. The association graph $G_{\mathrm{assc}} = (V_{\mathrm{assc}}, E_{\mathrm{assc}})$ is a description of the isomorphic relationships between the pairs of vertices in both graphs. We take a pair of vertices from G_1 and G_2, and if the pairs are isomorphic, the association graph has an edge denoting that relation. From the definition it follows that the association graph can be easily computed in $O(N^4)$ time. On the other hand, an association graph has N^2 vertices, so the worst case memory consumption will occur when the association graph is a complete graph, resulting in $|E_{\mathrm{assc}}| = N^4$. In practice, the memory consumption depends on the *edge density*. Edge density refers to the number of edges normalized by the number of vertices.

In the second step, the *maximum clique* (Diestel, 2000) is found from the association graph. A clique is a subgraph, which has edges between all its vertices. This step, however, takes $O(N^2 \cdot N^2!)$ time (Conte et al., 2007) and is the bottleneck of the algorithm. We therefore consider a faster heuristic known as *Reactive Local Search for Maximum Clique* (RLSMC) (Battiti and Mascia, 2007). It is an iterative local search heuristic, in which a new maximum clique solution is found in each iteration, taking into account the solutions from the previous iterations. One iteration of this algorithm works in $O(\max\{N^2, |E_{\mathrm{assc}}|\})$ time. Drawback of the heuristic is that there is no guarantee on the result compared to the exact algorithm. The steps of the matching algorithm are summarized in Algorithm 1.

Algorithm 1. Computation of D_{MCS}

Cluster the input data sets X_1 and X_2
Construct kNNG's G_1 and G_2 from the clustering result of X_1 and X_2
Construct association graph from G_1 and G_2
Find maximum clique from the association graph
Calculate score using $D_{\mathrm{MCS}}(G_1, G_2)$

2.4. Graph invariance to affine transformation

To show that the (1) is invariant to rotation, uniform scaling and translation, we need to show that kNN graph is invariant under these properties. If kNN graph is invariant, the corresponding maximum common subgraph is invariant also and the claim is true. Next we show that kNN graph is indeed invariant. Let $\boldsymbol{A}$ be a *rotation matrix* which includes uniform scaling by factor α, that is, $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^{\mathrm{T}} = \alpha\boldsymbol{I}$. Furthermore, let $f(\boldsymbol{x})$ be the affine map $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$. The squared Euclidean distance between two arbitrary vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ which both have been subjected to transformation f is,

$$\begin{aligned}
\|f(\boldsymbol{x}) - f(\boldsymbol{y})\|^2 &= \|(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}) - (\boldsymbol{A}\boldsymbol{y} + \boldsymbol{b})\|^2 = (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}\boldsymbol{y})^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}\boldsymbol{y}) \\
&= (\boldsymbol{A}\boldsymbol{x})^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}\boldsymbol{y}) - (\boldsymbol{A}\boldsymbol{y})^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}\boldsymbol{y}) \\
&= \boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{y} - \boldsymbol{y}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} + \boldsymbol{y}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{y} \\
&= \alpha(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{y}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{y}^{\mathrm{T}}\boldsymbol{y}) = \alpha\|\boldsymbol{x} - \boldsymbol{y}\|^2,
\end{aligned} \tag{2}$$

which is the distance in the original space multiplied by a constant. Therefore, the kNN neighborhood assignments in the transformed space do not change.

3. Hybrid graph- and template-based matching

Conventional speaker recognition methods based on vector quantization (VQ) or Gaussian mixture modeling (GMM) directly match the unknown person's feature vectors against the reference model(s). This *template*-based approach gives a real-valued match score but is not robust against affine transformations of features. The graph matching based on the MCS distance (1), on the other hand, implements *structural* matching as the match score is computed from the graph structures without any reference to the original vectors. The match score, however, is discrete and cannot be expected to be a very discriminative dissimilarity value: often no unique best-matching speaker was observed in our preliminary speaker identification experiments. We therefore formulate a hybrid approach that combines the good properties of the graph- and template-based approaches by using graph matching as an alignment tool prior to template matching.

The method has two main phases. In the first phase, the reference and test vector sets are paired by graph matching. The corresponding vectors from the isomorphic vertices from the kNN graphs are paired together. Denoting the N cluster centroids from the test and reference sets by $X = \{\boldsymbol{x}_i\},\ i = 1, \ldots, N$ and $Y = \{\boldsymbol{y}_i\}, \ldots, N$, respectively, graph matching produces the index pairs $(i_k, j_k),\ k = 1, 2, \ldots, K$ where $i_k, j_k \in \{1, 2, \ldots, N\}$ and $K = |\mathrm{MCS}(G_1, G_2)|$.

To this end, we have defined a pairing between the two sets of vectors that allows optimization under the affine distortion model. Assuming that $\widetilde{X} = \{\boldsymbol{x}_{i_k}\},\ k = 1, 2, \ldots, K$ contains the "clean" and $\widetilde{Y} = \{\boldsymbol{y}_{j_k}\},\ k = 1, 2, \ldots, K$ contains the "noisy" vectors, we want to

Fig. 3. Matching before (left) and after the alignment by MCS (right).

find the affine transformation that minimizes the squared error between the two paired vector sets,

$$(\boldsymbol{A}^*, \boldsymbol{b}^*) = \arg\min_{(\boldsymbol{A},\boldsymbol{b})} \sum_{k=1}^{K} \|\boldsymbol{y}_{j_k} - (\boldsymbol{A}\boldsymbol{x}_{i_k} + \boldsymbol{b})\|^2. \tag{3}$$

To solve the optimization problem (3), we represent the vectors in extended form by adding one more dimension with scalar 1, $\hat{\boldsymbol{x}} = [\boldsymbol{x}^{\mathrm{T}}, 1]^{\mathrm{T}}$. The extended vectors from each set are re-arranged into two matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ according to the pairing, that is, $\boldsymbol{X} = (\hat{\boldsymbol{x}}_{i_1}, \hat{\boldsymbol{x}}_{i_2}, \ldots, \hat{\boldsymbol{x}}_{i_K})$ and $\boldsymbol{Y} = (\hat{\boldsymbol{y}}_{j_1}, \hat{\boldsymbol{y}}_{j_2}, \ldots, \hat{\boldsymbol{y}}_{j_K})$. The parameters of the affine transformation, on the other hand, can be represented as

$$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{A} & & & & \boldsymbol{b} \\ 0 & \cdots & 0 & 0 & 1 \end{pmatrix}, \tag{4}$$

and the optimization problem (3) can be written as a system of linear equations:

$$\boldsymbol{Y} = \widehat{\boldsymbol{W}}\boldsymbol{X}, \tag{5}$$

which has the standard least squares solution via normal equations:

$$\widehat{\boldsymbol{W}} = \boldsymbol{Y}\boldsymbol{X}^{\mathrm{T}}(\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}})^{-1}. \tag{6}$$

The least squares approximation of the transform parameters can then be obtained directly from the matrix $\widehat{\boldsymbol{W}}$.

In the second phase, all the remaining (non-matched) code vectors from the sets $\overline{X} = X \setminus \widetilde{X}$ and $\overline{Y} = Y \setminus \widetilde{Y}$ are matched. Test vectors are first transformed using the affine transformation and then matched by the quantization distortion defined as:

$$D_{\mathrm{TSE}}(\overline{X}, \overline{Y}) = \sum_{\boldsymbol{x} \in \overline{X}} \min_{\boldsymbol{y} \in \overline{Y}} \|\boldsymbol{x} - \boldsymbol{y}\|^2. \tag{7}$$

The reason for excluding the MCS-paired vectors is that these vectors already match perfectly in the sense of the graph distance. A nonzero distortion value is obtained when the reference and test distribution shapes (in the sense of the graph structure) differ from each other. In this sense, the score can said to be a hybrid structural and template score. The hybrid matching algorithm is summarized in Algorithm 2, and illustrated in Fig. 3.

Algorithm 2. Hybrid graph- and template-based matching

Find the pairing (i_k, j_k), $k = 1, 2, \ldots, K$ using MCS
Find the solution $(\boldsymbol{A}^*, \boldsymbol{b}^*)$ to (3) using least squares
$\overline{X} \leftarrow X \setminus \{\boldsymbol{x}_{i_k}\}, k = 1, \ldots, K$
$\overline{Y} \leftarrow Y \setminus \{\boldsymbol{y}_{j_k}\}, k = 1, \ldots, K$
$\overline{X} \leftarrow$ replace each vector $\boldsymbol{x}$ in $\overline{X}$ by $\boldsymbol{A}^*\boldsymbol{x} + \boldsymbol{b}^*$
Compute the score as $D_{\mathrm{TSE}}(\overline{Y}, \overline{X}) + D_{\mathrm{TSE}}(\overline{X}, \overline{Y})$

4. Experiments

4.1. Effectiveness of the heuristic graph matching algorithm

We have two graph matching algorithms, the exact Durand–Pasari algorithm (exact MCS) and the heuristic RLSMC algorithm (heuristic MCS). With the exact algorithm, the size of the graph is restricted to 20 vertices in practise due to its exponential time complexity. This is clearly too small a number for modeling spectral features. Therefore, for any large scale experiments, the heuristic variant must be used.

We first study the quality of the heuristic algorithm by generating a random graph G with N vertices using the so-called *Erdős-Renyi* (Diestel, 2000) algorithm. We then randomly permutate the vertex labels in order to obtain a graph G' which is isomorphic to G. The result of the heuristic algorithm is compared to the known optimum (N), and the relative differences are summarized in Fig. 4 as a function of the iterations in the algorithm. For smaller graphs, the heuristic solution is close (within 15%) to the exact solution which is acceptable for our method. For the further experiments, we fix the number of iterations to 5000.

4.2. Demonstration of rotation invariance

Next, we demonstrate the rotation invariance property of the graph-based methods. We extract mel-frequency cepstral coefficients (MFCCs) from two speakers (see Section 4.3). We then rotate these feature sets with pre-specified rotation matrices and study whether the graph-based approaches are capable of undoing the rotation. In the ideal case, the match score (distortion) of 0 would be obtained regardless of rotation.

We refer to the two speakers as A and B, and denote the original codebooks by A0 and B0, respectively. Both of these sets are then rotated by 45° three times (in a sequence) by using different rotation axes. Rotations are performed subsequently after each other, yielding rotated feature sets denoted as A1, A2 and A3 (B1, B2 and B3, respectively). The first two dimensions of the feature sets are displayed in Figs. 5 and 6 along with $N = 20$ codebook centroids.

We match the dataset A0 against all the eight datasets and compute the match scores using the template-based method (TSE), the structural approach (graph matching), and the hybrid method (graph matching followed by TSE matching). For the graph matching, we use either the Durand–Pasari algorithm (exact) or the RLSMC algorithm (heuristic). The match scores (distances) are given in Table 1.

The TSE distance increases when the datasets become rotated more, as expected. The graph-based approach, on the other hand, is much less affected by rotation. The exact MCS variant gives a perfect match for the correct speaker in all cases. The heuristic MCS variant is slightly less accurate, but the order of the speaker scores is still correct: larger distortions are obtained for speaker B as should be. It is also noted that the hybrid approach gives a

Fig. 4. The effect of the number of iterations on the accuracy of the local search heuristic (RLSMC algorithm) relative to the exact algorithm (Durand–Pasari).

Fig. 5. The first two dimensions of the original A0 (left), along with the rotated sets A1, A2 and A3.

Fig. 6. The first two dimensions of the original feature set B0 (left), along with the rotated sets B1, B2 and B3.

Table 1
Match scores from the rotation experiment

Algorithm	Match A0 against							
	A0	A1	A2	A3	B0	B1	B2	B3
Template-based (TSE)	0.00	1.00	1.22	1.62	0.61	1.44	2.03	2.47
Structural (graph)								
Exact MCS	0.00	0.00	0.00	0.00	0.35	0.35	0.35	0.35
Heuristic MCS	0.05	0.05	0.05	0.05	0.35	0.35	0.35	0.35
Hybrid (graph + TSE)								
Exact MCS	0.00	0.00	0.00	0.00	1.35	2.25	2.85	2.43
Heuristic MCS	0.00	0.03	0.05	0.03	0.37	0.87	0.92	1.53

wider range of match scores compared with the purely structural matching as expected.

4.3. Speaker recognition experiments

We use a subset of the NIST-1999 SRE speaker recognition evaluation corpus for the speaker recognition experiments. The data is conversational speech acquired over a landline telephone network, and the sampling rate is 8 kHz. We use the 12 lowest mel-frequency cepstral coefficients (MFCCs) as the acoustic features, excluding the 0th coefficient. The window size is 30 milliseconds, and the mel-frequency filterbank consists of 27 triangular-shaped filters.

We have selected 30 male target speakers for our experiments from the training section of the corpus. In the original corpus, each speaker has two training files from two different recording sessions denoted as "*a*" and "*b*". We use the "*a*" files as the training files and the "*b*" files as the test data. Identification accuracy is measured by performing closed-set identification on these 30 speakers. For the verification experiments, we match all the (a, b) cross-pairs from these speakers, and an additional 50 genuine trials from additional speakers from the same corpus. This amounts to a total number of 80 genuine and 870 impostor trials. Verification accuracy is measured using the detection error trade-off (DET) plots and equal error rate (EER).

We include two standard approaches as reference systems: adapted GMM (Reynolds et al., 2000) and adapted VQ (Hautamäki et al., 2008). Both of them use a previously trained universal background model (UBM) to train the target-specific GMM, or a codebook using maximum a posteriori (MAP) criterion. We use 70 speakers from the other training segments of the NIST-1999 corpus to train the UBMs. Adaptation relevance factors of $r = 16$ and $r = 12$ are used for the GMM- and VQ-based models, respectively. Only the mean vectors are adapted in the GMM system (Reynolds et al., 2000). In the matching phase, the target score is normalized using the background model score.

The EERs are given in Table 2 and the DET plot for model size $N = 64$ is shown in Fig. 7. Overall, the graph-based approach is comparable to GMM and VQ. The graph-based approach is slightly better at small false alarm rates. The accuracy of GMM increases as a function of model size whereas the accuracy of the graph-based

Table 2
Speaker verification accuracies (EER %) on a subset of the NIST-1999 corpus

Model size	GMM baseline	VQ baseline	Graph + TSE (heuristic MCS)
32	15.3	12.5	20.0
64	17.1	12.5	13.8
128	17.2	12.5	15.0
256	15.0	11.3	N/A
512	13.8	12.5	N/A
1024	12.5	11.3	N/A

Model size refers to the number of Gaussians (GMM), code vectors (VQ) or graph vertices (graph + TSE).

Fig. 7. Verification accuracies of the methods on the NIST-1999 corpus.

Table 3
Speaker identification error rates on a subset of the NIST-1999 corpus

Model	Individual systems		Fusion		
	GMM-UBM (%)	Graph + TSE (%)	Score-level (%)	Rank-level (%)	Voting (%)
32	30	43	27	30	37
64	23	30	20	23	27
128	23	30	23	29	27

approach degrades at $N = 128$. A likely explanation is that the heuristic MCS algorithm has not converged and the number of iterations should be set higher since the size of the association graph is much larger compared with $N = 64$. For larger graphs, the space required by the association graph and the running times were impractically high.

Finally, we choose the GMM- and graph-based systems for a fusion experiment. In *score fusion*, we first normalize the scores of each classifier to have zero mean and unit variance, followed by equal weights summation. In *rank fusion*, the scores of each classifier are converted to rank values and summed. In *voting*, the ranks are hardened to binary decisions. In the case of a tie, we randomly choose between the best-matching speakers. The fusion results are given in Table 3. Score fusion improves accuracy for the model sizes $N = 32$ and $N = 64$, which suggests that the two approaches might contain mutually complementary information.

5. Conclusions

In this study, we have introduced graph-based matching approach to text-independent speaker recognition. The approach was motivated by the fact that a neighborhood graph encodes *structural* information about the feature space. Under the affine distortion model – including rotation, translation, and uniform scaling – ideally the neighborhood graph should not change.

The performance of the proposed method was comparable to the GMM- and VQ-based approaches. A fusion experiment demonstrated that GMM- and graph-based methods might contain mutually complementary information. The approach has potential to complement or replace currently used statistical and template-based methods.

The method, however, has several practical problems to be solved before it can be utilized in real-life speaker recognition systems. First, exact graph matching is computationally expensive, and heuristic algorithm needs to be used which weakens the performance. Second, the size of the association graph grows fast for large models, which implies increased running time. The largest graph that we could test in reasonable time was 128. A possible future solution could be based on a heuristic algorithm, which solves the graph matching problem directly, without reducing it first to the maximum clique search from the association graph. To further speed up scoring in the identification task, some form of decision tree in which the "feature points" represent tree nodes, could be used.

In the current approach, the "feature points" from the reference and test sets were found by clustering and implicitly assumed to correspond to phonetic classes. In general, the joint effects of channel and text (phonetic) mismatch are not well understood. Recently, excellent results have been obtained by using phone-class constrained GMMs which reduces text mismatch by phone recognition (Castaldo et al., 2007). The graph-based method could be used by restricting matching onto the same phone classes between training and test utterances.

Different graph structures are also possible. In this study, we considered unweighted kNN graph where a node is either connected or not to another node. A possible future direction would be using real-valued weights (such as Euclidean distances between points) and re-defining the matching framework for such graphs. Current likelihood-based (or frame-based) approaches also assume independence of the frames, largely ignoring utterance-level structural information. Graph matching could be potentially used as an alternative matching tool for the existing GMM-based systems. These are points for future research.

References

Auckenthaler, R., Carey, M., Lloyd-Thomas, H., 2000. Score normalization for text-independent speaker verification systems. Digital Signal Process. 10, 42–54.

Battiti, R., Mascia, F., 2007. Reactive local search for maximum clique: A new implementation. Technical Report DIT-07-018, Informatica e Telecomunicazioni, University of Trento.

Bunke, H., Messmer, B., 1997. Recent advances in graph matching. J. Pattern Recognition Artif. Intell. 11 (1), 169–203.

Bunke, H., Shearer, K., 1998. A graph distance metric based on the maximal common subgraph. Pattern Recognition Lett. 19 (3–4), 255–259.

Burge, M., Burger, W., 2000. Ear biometrics in computer vision. In: Internat. Conf. on Pattern Recognition (ICPR), vol. 2, Barcelona, Spain, September 2000, pp. 822–826.

Castaldo, F., Colibro, D., Dalmasso, E., Laface, P., Vair, C., 2007. Compensation of nuisance factors for speaker and language recognition. IEEE Trans. Audio Speech Language Process. 15 (7), 1969–1978.

Conte, D., Foggia, P., Vento, M., 2007. Challenging complexity of maximum common subgraph detection algorithms: A performance analysis of three algorithms on a wide database of graphs. J. Graph Algorithms Appl. 11 (1), 99–143.

Diestel, R., 2000. Graph Theory, second ed. Springer-Verlag, New York.

Durand, P., Pasari, R., Tsai, C.C., 1999. An efficient algorithm for similarity analysis of molecules. Internet J. Chem. 2 (17).

Fränti, P., Virmajoki, O., Hautamäki, V., 2006. Fast agglomerative clustering using a *k*-nearest neighbor graph. IEEE Trans. Pattern Anal. Machine Intell. 28 (11), 1875–1881.

Hautamäki, V., Kinnunen, T., Kärkkäinen, I., Saastamoinen, J., Tuononen, M., Fränti, P., 2008. Maximum a posteriori adaptation of the centroid model for speaker verification. IEEE Signal Process. Lett. 15, 162–165.

Kann, V., 1992. On the approximability of the maximum common subgraph problem. In: Proc. 9th Annual Symp. on Theoretical Aspects Computer Science (STACS), Cachan, France, February, pp. 377–388.

Kenny, P., Boulianne, G., Ouellet, P., Dumouchel, P., 2007. Speaker and session variability in GMM-based speaker verification. IEEE Trans. Audio Speech Language Process. 15 (4), 1448–1460.

Levi, G., 1972. A note on the derivation of maximal common subgraphs of two directed or undirected graphs. Calcolo 9, 341–352.

MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations. In: Proc. 5th Berkeley Symp. on Mathematical Statistics and Probabilit, vol. I, University of California, pp. 281–297.

Mak, M.-W., Tsang, C.-L., 2004. Stochastic feature transformation with divergence-based out-of-handset rejection for robust speaker verification. EURASIP J. Appl. Signal Process. 4, 452–465.

Mammone, R.J., Zhang, X., Ramachandran, R.P., 1996. Robust speaker recognition: A feature-based approach. IEEE Signal Process. Mag. 13 (5), 58–71.

Pelecanos, J., Sridharan, S., 2001. Feature warping for robust speaker verification. In: Proc. Speaker Odyssey, Crete, Greece, pp. 213–218.

Reynolds, D., Quatieri, T., Dunn, R., 2000. Speaker verification using adapted gaussian mixture models. Digital Signal Process. 10 (1), 19–41.

Siohan, O., Lee, C.-H., 1997. Iterative noise and channel estimation under the stochastic matching algorithm framework. IEEE Signal Process. Lett. 4 (11), 304–306.

Teunen, R., Shahshahani, B., Heck, L., 2002. A model-based transformational approach to robust speaker recognition. In: Proc. ICSLP 2000, vol. 2, Beijing, China, pp. 495–498.

Vogt, R., Sridharan, S., 2008. Explicit modeling of session variability for speaker verification. Comput. Speech Language 22 (1), 17–38.

6

Publication P6

T. Kinnunen, V. Hautamäki and P. Fränti, "On the use of long-term average spectrum in automatic speaker recognition", in *Proceedings of the 5th International Symposium on Chinese Spoken Language Processing* (ISCSLP 2006), Vol. 2, pp. 559-567, Singapore, December 2006.

Copyright © 2006 COLIPS. Reprinted with permission.

On the Use of Long-Term Average Spectrum in Automatic Speaker Recognition

Tomi Kinnunen[1], Ville Hautamäki[2], and Pasi Fränti[2]

[1] Speech and Dialogue Processing Lab
Institute for Infocomm Research (I^2R)
21 Heng Mui Keng Terrace, Singapore 119613
ktomi@i2r.a-star.edu.sg
[2] Speech and Image Processing Unit
Department of Computer Science, University of Joensuu
P.O. Box 111, FIN-80101 Joensuu, Finland
{villeh, franti}@cs.joensuu.fi

Abstract. State-of-the-art automatic speaker recognition systems use mel-frequency cepstral coefficients (MFCC) features to describe the spectral properties of speakers. In forensic phonetics, the long-term average spectrum (LTAS) has been used for the same purpose. LTAS provides an intuitive graphical representation which can be used to visualize and quantify speaker differences. However, few studies have reported the use of LTAS in automatic speaker recognition. Thus, the purpose of this paper is to systematically study how to use the LTAS in automatic speaker recognition. We will also find out whether it provides additional discriminative information in respect to the MFCC-based system.

1 Introduction

Differences in our voices arise from both *physical* factors (anatomy), and *behavioral* factors (the way of speaking). Both of these factors give rise to several measurable quantities that can be used as features in speaker recognition. In state-of-the-art automatic speaker recognition systems, multiple features are used in parallel to complement each other. In this study, we focus on spectral features because they give the best accuracy among several high- and low-level features [1].

In automatic speaker recognition, spectral features are computed from short frames (20-40 milliseconds) with the rate of 50-100 frames per second. The most commonly employed features are *mel-frequency cepstral coefficients* (MFCC) [2], appended with their first and second order delta coefficients. The short-term feature computation is followed by statistical modeling of the distribution of the vectors; each speaker produces a characteristic "cloud" in the feature space. The state-of-the-art model is the *Gaussian mixture model* (GMM) [3]. In GMM, the "feature cloud" is modeled by fitting a finite set (256-2048) of Gaussian distributions to the training data so that they characterize the data as well as possible.

Fig. 1. Examples of LTAS computed from NIST-2001 corpus (window length = 50 ms, frequency spacing = 16 Hz).

There might be a simpler and computationally more efficient way than MFCC + GMM to describe the spectral characteristics of a speaker. In forensic phonetics [4], one approach to describe the resonance characteristics of a speaker is *long-term average spectrum* (LTAS). It is computed by time-averaging the short-term Fourier magnitude spectra, resulting in one feature vector for the whole speech sample (see Fig. 1).

The advantage of LTAS from a *forensic* perspective is that it has more or less direct physical interpretation, relating to the location of the vocal tract resonances. This makes LTAS more justified as an evidence than MFCC coefficients, which do not have direct phonetic interpretation. LTAS vectors of the questioned speech sample and the suspects speech sample can be plotted on top of each other for visual verification of the degree of similarity [5]. LTAS and other features can be complemented by auditory analysis and (semi-)automatic methods.

The advantages of LTAS from *automatic* speaker recognition perspective would be simple implementation and computational efficiency compared with the GMM. In particular, there is no separate training phase included; the extracted LTAS vector will be used as the speaker model directly and matched with the test utterance LTAS using a distance measure.

This study has two main objectives. First, although LTAS is used in forensic casework, we are not aware of systematic studies reporting the effect of the control parameters. LTAS is affected by changes in channel conditions, and robust matching and score normalization are important when LTAS is considered for telephony speaker recognition. Thus, the first goal of this paper is to study the parameters of LTAS extraction, matching, and score normalization.

The second objective of the study is to find out the usefulness of LTAS in automatic recognition. In particular, we want to answer the following questions:

- How does recognition accuracy of LTAS compare with MFCC+GMM?
- How does computational cost of LTAS compare with MFCC+GMM?

– Can LTAS and MFCC+GMM be fused for improved accuracy?
– Is there any reason to use LTAS in automatic recognition?

The rest of the paper is organized as follows. In Section 2, we describe the computation and matching of LTAS. Section 3 gives the details of the datasets used, and the experimental results are given in Section 4. Finally, conclusions are drawn in Section 5.

2 Computation and Matching of LTAS

From a signal processing point of view, LTAS computation belongs to the class of *power spectral density* (PSD) [6] estimation methods. We consider two alternative methods for computing LTAS. The first one is based on a single transformation followed by spectrum size reduction, and the second one is based on time-averaging of the short-term spectra.

In the *single-transformation LTAS*, we first compute a single fast Fourier transform (FFT) over the whole signal. This is preceded by Hamming windowing and zero padding to the length of the next power of two. Note that the length of the spectrum varies depending on the input. To get a fixed-length LTAS vector, the number of power spectrum points is reduced by averaging neighboring frequency bins, which corresponds applying a uniformly spaced filterbank. The single-transformation method is used, for instance, in the open-source *Praat*[3] speech analysis program, and it will be used here as a reference method.

Another method to compute LTAS is to divide the signal into overlapping frames, compute the power spectrum of each frame, and to average the spectra. As in the single-transformation LTAS, we apply Hamming windowing, and set the FFT size to the next power of two of the frame length. The short-term averaging method is also known as *Welch's method* [7], and it is better suited for practical applications.

The LTAS vector elements are nonlinearly compressed by applying logarithm. Log-compression balances the spectrum, and, according to our experiments, it systematically outperforms the LTAS represented in linear amplitude scale.

Finally, we need a distance measure between two LTAS vectors. We consider four simple distance measures: Euclidean distance, correlation coefficient, cosine measure and *Kullback-Leibler* (KL) divergence [8]. For the KL divergence, the LTAS vector is considered as a probability mass function. In order to meet the probabilistic constraints, the vector elements are translated to positive values, followed by normalization so that the elements sum to 1.

To increase robustness against acoustic mismatch, the raw match score is normalized by other speakers' scores. For this, we apply the *test normalization* ("T-norm") method [9]. The unknown LTAS vector is matched against a set of pseudoimpostor models (*T-norm cohort*), and the mean and standard deviation are obtained. The normalized match score is obtained from the raw score by subtracting the mean and by dividing by the standard deviation.

[3] `http://www.praat.org`

3 Experimental Setup

We use the NIST-1999 and NIST-2001 speaker recognition benchmarking corpora for our experiments [10]. Both corpora consists of conversational telephony data, NIST-1999 being recorded over the landline network and NIST-2001 over the cellular network. The NIST-1999 corpus is used for studying the effect of the feature extraction parameters, and comparing the distance measures. The NIST-2001 corpus is used for validating the results, studying score normalization, and comparing the accuracy and time consumption with the standard MFCC+GMM recognizer [3].

We use the training files of the male speakers of the NIST-1999 corpus for parameter tuning. This subset consists of 230 speakers, each represented by two audio files labeled as "a" and "b". Both samples have a duration of one minute. We consider the "a" samples as the reference samples, and the "b" samples as the unknown ones. We report both the identification and verification accuracies. We use closed-set identification error rate (IER) to measure the identification accuracy, and equal error rate (EER) to measure the verification accuracy. Equal error rate corresponds to the verification threshold for which the false acceptance and the false rejection rates are equal.

For the NIST-2001 corpus, we report verification accuracy on the 1-speaker detection task. The detection list provided by NIST consists of 9350 male trials (850 genuine + 8500 impostor) and 13068 female trials (1188 genuine + 11880 impostor). There are two minutes of training data per speaker, and the length of the test segments varies from a few seconds up to one minute. The NIST-2001 development set consisting of 60 speakers is used as the T-norm cohort set.

For the MFCC+GMM recognizer, we use the MFCC coefficients 1-12, appended with the delta and double-delta coefficients. Utterance-level mean subtraction and variance normalization are applied to the features. The universal background model of 512 components is trained from the NIST-2001 development set. The target speaker models are trained by adjusting the background model mean vectors towards the speaker's training data, see details in [3]. The fast C-top scoring algorithm described in [3] is used for matching.

4 Results

4.1 Summary of the Tuning Results

For the single-transformation LTAS, the number of FFT bins was varied in powers of two between 32-2048. For the short-term averaged LTAS, the frame length was varied between 30-320 milliseconds with a step of 10 milliseconds, and the window overlap was fixed to 50%. Table 1 summarizes the best, the worst and the average (mean $\pm$ standard deviation) accuracies. For completeness, verification accuracies of the single and short-term methods are compared in the detection error tradeoff (DET) curve of Fig. 2.

We observe that the single-transform and short-term variants are equally good. For instance, Fig. 2 shows that the short-term variant outperforms the

Table 1. Results for the tuning set.

	Eucl.	Corr.	Cos.	KL dist.
Best				
EER (single) (%)	30.0 (64 bins)	30.9 (64 bins)	18.3 (128 bins)	**18.2** (128 bins)
EER (short-term) (%)	20.4 (120 ms)	20.4 (400 ms)	19.6 (170 ms)	**18.2** (190 ms)
IER (single) (%)	76.1 (512 bins)	54.8 (512 bins)	**48.7** (128 bins)	**48.7** (128 bins)
IER (short-term) (%)	52.6 (40 ms)	**45.2** (50 ms)	47.8 (50 ms)	47.0 (4000 ms)
Average				
EER (single) (%)	31.8±1.3	22.2±1.0	**18.7**±0.5	**18.7**±0.5
EER (short-term) (%)	21.3±0.5	21.2±0.3	20.3±0.4	**19.2**±0.5
IER (single) (%)	77.8±1.9	57.1±3.3	**51.4**±3.1	**51.4**±3.1
IER (short-term) (%)	58.4±1.8	**47.8**±1.4	49.8±1.1	50.2±1.7
Worst				
EER (single) (%)	32.8 (256 bins)	23.5 (32 bins)	**19.6** (2048 bins)	**19.6** (2048 bins)
EER (short-term) (%)	22.2 (320 ms)	21.4 (110 ms)	21.2 (50 ms)	**20.0** (80 ms)
IER (single) (%)	80.9 (32 bins)	63.9 (32 bins)	**58.3** (32 bins)	**58.3** (32 bins)
IER (short-term) (%)	60.9 (200 ms)	**47.8** (250 ms)	51.0 (280 ms)	53.0 (30 ms)

single-transformation variant for low false acceptance rate (secure end) of the DET curve but the situation is reversed for low false rejection rate (user-convenience end). The equal error rates are close to each other.

Fig. 2. Comparison of single-transformation and short-term variants for LTAS computation (KL divergence).

4.2 T-norm and Comparison with MFCC+GMM

Next, we validate our results using the NIST-2001 evaluation set. We use the short-term averaging variant with 200 millisecond window to compute LTAS. The verification results with and without score normalization are given in Table

2. The error rates are higher than for NIST-1999, for which the likely reason is that test segments are in general much shorter and trials include channel missmatch for the NIST-2001. Score normalization improves accuracy in all cases as expected. However, the KL measure does not give the best result as opposed to the NIST-1999 results. The reason for this is unknown.

Table 2. Equal error rates (%) for the NIST-2001 corpus.

Normalization	Eucl.	Corr.	Cos.	Kullb.-Leib.
None	31.7	28.0	27.2	31.7
T-norm	30.4	24.2	24.9	29.0

Next, we study the effect of channel mismatch to LTAS and give the MFCC + GMM baseline as a reference. To study the effect of channel mismatch, NIST-2001 detection list was divided into "matched" and "mismatched" trials. The matched and mismatched trials refer to the trials with same and different phone model with the training and test files, respectively.

The left and right panels of Fig. 3 show the DET curves for the matched and mismatched channel cases, respectively. Not surprisingly, channel mismatch degrades the accuracy of both the LTAS and the MFCC+GMM recognizers. The MFCC+GMM recognizer outperforms LTAS without a doubt, as expected. Note that, because of computational reasons, we did not apply T-norm for the MFCC+GMM system. The MFCC+GMM error rate would be expected to decrease further by including T-norm.

Fig. 3. Verification results for the NIST-2001 corpus, matched channel type (left), mismatched channel type (right).

4.3 Time Consumption

Next, we study the computation times of LTAS and MFCC+GMM. All the experiments are carried out in 3GHz Intel Pentium 4 with 1024 MB of memory. All algorithms were implemented and run in Matlab 7. Tests were performed by first enrolling all speakers into a database and then perfoming the NIST-2001 evaluation protocol on the enrolled speakers. Running times are reported in seconds averaged over all test cases.

The speaker enrollment times are summarized in the Table 3. The running times of the single-transformation and short-term variants are practically the same, and LTAS is about 13 times faster than the MFCC+GMM recognizer.

Verification times are summarized in Table 4. Overall matching time of LTAS without score normalization is about 10 times faster than that of the MFCC + GMM. Adding score normalization increases the processing time of LTAS, and the baseline MFCC+GMM matching is faster than LTAS + T-norm. However, even with score normalization, overall processing time of LTAS is smaller, which is due to much faster feature extraction.

Table 3. Comparison of CPU times for enrollment (seconds).

	Feature extraction	Modeling	Total
single-transf. LTAS	1.0±0.0	-	1.0
short-term avg. LTAS	0.9±0.1	-	0.9
MFCC+GMM	9.2±1.1	4.4±0.1	13.6

For identification performance, the matching times should be multiplied by the number of speakers enrolled in the database. For example, identification with the short LTAS would take on average 0.2 + 0.1 = 0.3 seconds and with the MFCC+GMM system 2.6 + 104.4 = 107.0 seconds. Thus, there is a remarkable difference in the processing time required.

Table 4. Comparison of CPU times for verification (seconds).

	Feature extraction	Matching	Total
single-transf. LTAS	0.3±0.1	< 0.01	0.3
single-transf. LTAS+T-norm	0.3±0.1	1.8±0.2	2.1
short-term avg. LTAS	0.2±0.1	< 0.01	0.2
short-term avg. LTAS+T-norm	0.2±0.1	1.8±0.2	2.0
MFCC+GMM	2.6±1.1	0.6±0.9	3.2

4.4 Combining LTAS and MFCC+GMM

Finally, we want to find out whether it is advantageous to combine LTAS and MFCC+GMM recognizers. We use weighted sum to combine the classifier output

scores so that $s_{\text{fused}} = w \cdot s_{\text{MFCC+GMM}} + (1 - w) \cdot s_{\text{LTAS}}$, where $0 \leq w \leq 1$. Here, $s_{\text{MFCC+GMM}}$ is the average log likelihood ratio from the MFCC+GMM recognizer, and s_{LTAS} is the T-normalized correlation coefficient from the LTAS recognizer. The EER as a function of w and the DET curve for $w = 0.96$ are shown in Fig. 4.

Fig. 4. ERR as a function of fusion weight (left) and Fusion results (right).

Combining LTAS and MFCC+GMM gives a slight improvement to the MFCC + GMM baseline over all detection thresholds. However, according to Fig. 4, the weight selection is critical; for this corpus, the best result is obtained in the range $[0.94 - 0.97]$, and this is likely to be different for other corpus. Moreover, as the relative gain of combining LTAS with MFCC+GMM is only marginal, we conclude that it is not worth combining these two recognizers.

5 Conclusions

In this paper, we have studied the use of long-term average spectrum feature for automatic speaker recognition. We compared two different methods for computing LTAS, a single-transformation variant and a short-term averaging variant. We also compared the LTAS performance with the baseline MFCC+GMM system, and attempted to combine the two recognizers.

Our experiments indicate that the accuracy and computational load of the single-transformation and the short-term averaging variants are practically the same. However, from the memory and real-time processing consideration viewpoints, the short-term averaging variant would be the recommended method.

The current study suggest that LTAS does not bring improvement to the standard MFCC+GMM configuration. However, the method is trivial to implement and it is computationally efficient. One possible application in automatic recognition could be speeding up speaker identification from a large database [11]. For instance, LTAS could be used to prune out speakers who have a very

large distance from the unknown sample. After this, the remaining candidate speakers could be scored more accurately by the MFCC+GMM recognizer.

To sum up, we conclude that LTAS has little use in automatic speaker recognition if the recognition accuracy is the only motivation.

References

1. Reynolds, D., Andrews, W., Campbell, J., Navratil, J., Peskin, B., Adami, A., Jin, Q., Klusacek, D., Abramson, J., Mihaescu, R., Godfrey, J., Jones, D., Xiang, B.: The SuperSID project: exploiting high-level information for high-accuracy speaker recognition. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP 2003), Hong Kong (2003) 784–787
2. Huang, X., Acero, A., Hon, H.W.: Spoken Language Processing: a Guide to Theory, Algorithm, and System Development. Prentice-Hall, New Jersey (2001)
3. Reynolds, D., Quatieri, T., Dunn, R.: Speaker verification using adapted gaussian mixture models. Digital Signal Processing **10**(1) (2000) 19–41
4. Rose, P.: Forensic Speaker Identification. Taylor & Francis, London (2002)
5. Lindh, J.: Visual acoustic vs. aural perceptual speaker identification in a closed set of disguised voices. In: Proc. The 18th Swedish Phonetics Conference (FONETIK 2005), Göteborg, Sweden (2005) 17–20
6. Gray, R., Davisson, L.: An Introduction to Statistical Signal Processing. Cambridge University Press, Cambridge, United Kingdom (2003)
7. Welch, P.D.: The use of fast fourier transforms for the estimation of power spectra: A method based on time averaging over short modified periodograms. IEEE Transactions on Audio and Electroacoustics **15** (1967) 70–73
8. Cover, T., Thomas, J.: Elements of Information Theory. Wiley Interscience (1991)
9. Auckenthaler, R., Carey, M., Lloyd-Thomas, H.: Score normalization for text-independent speaker verification systems. Digital Signal Processing **10** (2000) 42–54
10. NIST: NIST speaker recognition evaluation. www page (2006) `http://www.nist.gov/speech/tests/spk/`.
11. Kinnunen, T., Karpov, E., Fränti, P.: Real-time speaker identification and verification. IEEE Trans. Audio, Speech, and Language Processing **14**(1) (2006) 277–288

7

Publication P7

V. Hautamäki, M. Tuononen, Tuija Niemi-Laitinen and P. Fränti, "Improving speaker verification by periodicity based voice activity detection", in *Proceedings of the 12th International Conference on Speech and Computer* (SPECOM 2007), Vol. 2, pp. 645-650, Moscow, October 2007.

Improving Speaker Verification by Periodicity Based Voice Activity Detection

Ville Hautamäki, Marko Tuononen, Tuija Niemi-Laitinen and Pasi Fränti

Speech and Image Processing Unit, Department of Computer Science and Statistics,
University of Joensuu, Finland

Abstract

We propose an improvement to a realtime speaker recognition system, where matching is performed instantly while subject is still speaking to the microphone. We present a voice activity detection (VAD) based on realtime periodicity analysis. Performance of the proposed method is compared against two existing methods: a realtime method based on long-term spectral divergence (LTSD) and a simple energy based method, which needs two passes on the data. The periodicity-based method clearly outperforms the other realtime method (LTSD), and performs comparably with the energy-based method when applied for NIST 2001 and 2006 speaker recognition evaluation corpora. The method is also tested for segmenting surveillance recordings, voice dialog and forensic applications.

1. Introduction

Voice Activity Detection (VAD) [1] aims at classifying a given sound frame as a speech or non-speech as. It is often used as a front-end component in voice-based applications such as automatic speech recognition, speech enhancement [2] and voice biometric [3]. A common property of these applications is that only human sounds (typically only speech) are of interest to the system, and it should therefore be separated from the background. Although VAD is widely needed and reasonably well-defined, existing solutions do not satisfy all the user requirements. The methods either must be trained for the particular application and conditions, or they can be highly unreliable when the conditions change. Demands for working solutions, however, are frequently requested by practitioners.

Traditionally, VAD research has been driven by telecommunications and voice-coding [1,2,4] applications, in which VAD has to operate with as small delay as possible and all the speech frames should be detected. Typically, these voice activity detectors work by modeling the noise signal statistics. Initial noise estimates are usually obtained from the beginning of the signal, which is updated as VAD makes non-speech decisions.

Even though realtime VAD for telecommunications application is maybe the most common, other applications also exist. For those applications, design criteria are usually different than telecom VAD's. In speech segmentation application, realtime operation is not as important as the goal is to process the input speech file (usually hours or even days long) in a background process and then the output will be used for the retrieval or the diarization tasks [5]. Segmentation of the forensic wiretapping is especially difficult task. In those recordings, no close talking microphone is used and the noise does not possess properties assumed in the telecom VAD's (long-term stationarity) [6].

In this study, our primary goal is to use VAD as a preprocessor for the realtime speaker verification. It is clear that including non-speech frames in the modeling process would bias the resulting model, especially if the number of non-speech frame is significant. It is also known that not all speech frames have equal discriminative power as it is well known that voiced phonemes are more discriminative than unvoiced phonemes [7], and therefore, it can be beneficial to drop out unvoiced frames to increase recognition. This is in contrast to telecom or speech recognition application where speech should be accurately detected.

In realtime speaker verification [8], we need to start processing speech frames with as small delay as possible. New recorded speech frame is pushed to signal processing subsystem,

and to scoring subsystem while subject keeps speaking to the microphone. Not only is this useful feature in the standard applications, but it is essential in low power mobile phones, where realtime processing is only option [9]. Good results in speaker verification can be achieved by a simple energy based VAD, but the method needs analyze the whole utterance before it can start to make speech/non-speech decisions [10].

Fig. 1. Example of the proposed system.

Our proposed system is based on the detecting the periodicity of the given frame. In contrast to the telecom voice activity detectors we do not model noise, but we base our decisions on the feature that is known to be short term stationary. Periodicity information has been used before in other speech technology applications, for example speech logging [11]. Our goal is to enhance the speaker verification performance.

2. VAD in Speaker Verification

In the speaker verification, the unknown speech sample is introduced to the system accompanied by a claim. The problem is considered to be a testing hypothesis that claimed person is the author of the utterance. In practice, we calculate *mel frequency cepstral coefficents* MFCC for each frame, which are then fed into either Gaussian mixture modeling algorithm (in training) or to the scoring algorithm (in testing).

We used the adapted Gaussian mixture model [3], in which the target speaker models are trained by adjusting the parameters of a universal background model (UBM) towards the speaker's training data. We used a diagonal covariance matrix GMM, which is trained separately for females and males using expectation maximization (EM) algorithm. EM is initialized by a variant of split algorithm and fine-tuned by K-means [12]. In testing phase, we used the average log-likelihood ratio to calculate a match score. Gender information was used to create separate UBM models for male and female speakers as defined in NIST corpus.

2.1. Periodicity based VAD

Given an accurate periodicity detector, we could voiced phonemes from the speech signal. In practice, we make speech/non-speech decisions on the frame level, so we need to consider how periodic a given frame is. We assume that noise is aperiodic, and therefore (in principle) we can detect speech parts of the utterance by thresholding the periodicity estimate. Although this is not generally true, it is expected that a significant part of speech can be separated from background noise in this way.

We used the periodicity detector defined in the YIN pitch estimation algorithm [13], which is an autocorrelation based algorithm. Autocorrelation function matches lagged version of the same frame on top of the frame itself. When the lagged version matches well to the current frame, we have found a period in the signal, where a pitch estimate can be extracted. Unfortunately, autocorrelation method itself is not very robust in estimation of pitch period because of the influence of the formants, especially the first formant. A complicated post-processing system is implemented in the YIN algorithm as follows.

First, the idea of autocorrelation is realized as the squared difference function, where function value is calculated at different lags:

$$d(\tau) = \sum_{j=1}^{W} (x_j - x_{j+\tau})^2 \,. \tag{1}$$

The minima of $d(\tau)$ will give estimate of aperiodicity and the corresponding τ can be turned into F0 estimate. Next, quadratic function is fitted to each local minima of the $d(\tau)$'s, and the result of the interpolation is then used as the aperiodicity estimate. Each calculated $d(\tau)$ is normalized by cumulative sum of the previous lags. Before normalization we initialize the $d(0) = 1$. Cumulative normalization is performed as

$$d'(\tau) = \frac{d(\tau)}{\frac{1}{\tau}\sum_{j=1}^{\tau} d(j)} \,. \tag{2}$$

$d'(\tau)$ values are evaluated from smallest τ to largest, while applying the user selected threshold to $d'(\tau)$'s. Such $d'(\tau)$ value is selected that is local minimum and smaller than the threshold.

The $d'(\tau)$ now represents the aperiodicity estimate of the frame, where 0 meaning periodic frame and 1 totally aperiodic frame. We obtain the final periodicity estimate p_i for a frame *i* by $p_i = 1 - d'(\tau)$. As the estimate is very noisy, we need to smooth it by averaging by the sliding window. See Fig. 1 for an example, where upper part contains speech waveform from NIST 2005 evaluation corpus, middle part its raw periodicity and lower part smoothed periodicity by window of length five.

2.2. Energy-based VAD

A simple energy-based approach is often used for voice activity detection. The approach in [10] was used in NIST 2006 evaluation: it measures the intra frame energy by calculating standard deviation of the frame and compares it to the expected SNR value (30 dB by default). A segment is defined as speech segment if it is within this marginal, and if it exceeds a given minimum energy level (-55 dB). This can be easily checked and it has found to work well for its purpose in voice biometric.

In more detail simple energy-based works as follows. First we calculate the logarithm compressed standard deviation of each frame:

$$S_i = 20\log_{10}\sqrt{\frac{1}{N-1}\sum_{j=1}^{W}\left(x_j^i - \bar{x}^i\right)^2} \,, \tag{3}$$

where x_j^i is the *j*th sample value of *i*th frame and $\bar{x}^i$ is the sample average of *i*th frame. Then frame *i* is detected as speech if $S_i > (\max_j S_j - T)$ and $S_i > -55$.

2.3. Long Term Spectral Divergence VAD

This approach measures long term spectral divergence (LTSD) between speech and noise. It formulates the speech/non-speech decision rule by comparing the long term spectral envelope to the average noise spectrum. It is also reasonably simple to implement and rather efficient compared to other VAD methods according to the experiments made in [2].

The algorithm operates on a window of $2M+1$ frames, where decision is made on a center frame. $\mathrm{LTSE}_M(k,l)$ is a short term spectral envelope, and it is calculated as follows:

$$\mathrm{LTSE}_M(k,l) = \max_{j=-M}^{M} X(k, l+j) \,, \tag{4}$$

where $X(k,l)$ is the magnitude of the *k*th spectral bin of the *l*th frame. Final decision is performed by thresholding $\mathrm{LTSD}_M(l)$ function:

$$\mathrm{LTSD}_M(l) = 10\log_{10}\left(\frac{1}{\mathrm{NS}}\sum_{k=0}^{\mathrm{NS}-1}\frac{\mathrm{LTSE}^2(k,l)}{\mathrm{N}^2(k)}\right), \tag{5}$$

where $\mathrm{N}(k)$ is the estimated noise magnitude spectrum of the *k*th bin and NS is the number of spectral bins. Noise estimate is updated on the fly when a negative decision has been made. Even though this strategy can account for long term changes in the noise characteristics, it can also bias the estimate and therefore, decrease the detection performance.

3. Speaker Verification Experiments

Feature extraction parameters for speaker verification experiments were set as follows. For the MFCC features, we use the coefficients 1-12, computed from a 27-channel mel-filterbank. The frame length is set to 30 milliseconds, with 33 % overlap. The MFCC vector is appended with its delta and double-delta coefficients at the frame level, yielding 36-dimensional data. Each feature is normalized by subtracting the mean and dividing by the standard deviation estimated from the file.

Voice activity detection parameters are tuned on the NIST 2001 evaluation corpora, in such a way that using one set of VAD parameters all audio files (trial, model and UBM) are segmented during the feature extraction. For periodicity VAD was tried window size parameters 3, 5 and 7. Threshold we varied between 0.60 and 0.90. LTSD and Energy based VAD's were compared by varying their decision thresholds.

In Fig. 2, periodicity VAD parameters are tuned with respect to NIST 2001 evaluation protocol. We can notice that smoothing window size should be small, and the threshold should be relatively low (less than 0.70), which leads to less frames being dropped out. We also notice that 512 size GMM outperforms 64 size model systematically, the difference is only 1% unit.

In Fig. 3, tuning results for energy and LTSD VADs are shown. We notice that LTSD has a sharp minimum around threshold value 40. Energy based VAD has a little bit larger minimum around threshold values of 34-39. Energy based VAD is cleary more stable with respect to its threshold than the LTSD VAD.

Fig. 2. NIST 2001 results for the proposed method with the model size 64 (left) and the 512 (right).

Fig. 3. NIST 2001 tuning results for the Energy (left) and LTSD (right) VAD's.

DET plots of the best tuning results are shown in Fig. 4. It is clear that all methods provide significantly better results than not using VAD at all. Both energy based and periodicity VADs clearly outperforms LTSD for NIST 2001. Periodicity VAD is almost one percent unit better than energy based VAD.

Fig. 4. NIST 2001 best results for the model size 512

Table 1. Summary of speaker verification results (% EER)

	NIST 2001 model 512		NIST 2001 model 64		NIST model
	EER	Thr	EER	Thr	EER
No VAD	13.63		16.00		44.39
LTSD	12.41	40	13.74	45	35.82
Energy	9.26	36	10.40	35	**16.63**
Periodicity	**8.46**	0.61	**9.58**	0.66	16.76

Summary of the speaker verification results with the corresponding VAD thresholds is shown in Table 1. Best parameters found in tuning with NIST 2001 corpus have been used for the NIST 2006 experiments. We obtained significantly higher error rates for NIST 2006 than NIST 2001, as expected. We also notice that for this corpus the energy based VAD slightly outperforms the periodicity VAD.

4. Speech Segmentation Experiments

Speech segmentation is a task of segmenting the audio signal in time to alternating speech and non-speech blocks. The quality of speech segmentation is evaluated by two measures: *false acceptance* (FA) and *false rejection* (FR) rates.

We used three different materials in our speech segmentation experiments: subset of the 2005 NIST Speaker Recognition Evaluation corpus, Bus-Stop timetable system [14] recordings, and material recorded in our lab [6].

Final comparative results are obtained in the Table 2. We note that the periodicity method is a clear winner in the Lab recording, and provides comparable results to simple energy based method for NIST05 dataset. On the other hand, for the Bus-stop material, periodicity method does not seem to perform adequately. One reason for the higher EER is that periodicity method detects as speech DTMF sounds, which are periodic, but are defined as noise in the ground truth.

Original LTSD VAD assumes that the beginning of the sound file contains noise that is typical to the sessions in question. We noted the poor segmentation results, so we collected non-speech segments from NIST05 evaluation corpus not used in the experiments and trained a separate noise model. This model was then used in all LTSD with trained noise experiments. LTSD VAD seems unstable with respect to the noise modeling scheme used.

Table 2. Summary of segmentation results (% EER).

	Bus-stop	Lab	NIST05
LTSD adaptive	19	14	40
LTSD trained	**6**	15	**1**
Energy	15	17	2

5. Conclusions

Best results are obtained either by energy-based method off-line, or by the proposed periodicity method on-line. In future work, we consider to combine the two measures in a single real-time method either by two-step thresholding, or by a simple classifier fusion.

6. Acknowledgements

The work was supported by the National Technology Agency of Finland (TEKES) as the projects New Methods and Applications of Speech Technology (PUMS).

References

1. *B. A, E. Schlomot*, *H. Su*, ``ITU-T recommendation g729 annex b: A silence compression scheme for use with g729 optimized for v.70 digital simultaneous voice and data applications," IEEE Communications Magazine, vol. 35, pp. 64-73, 1997.

2. *J. Ramírez*, *J. Segura*, *C. Benítez*, *A. de la Torre*, *A. Rubio*, ``Efficient voice activity detection algorithms using long-term speech information," Speech Communication, vol. 42, pp. 271-287, 2004.

3. *D. Reynolds*, *T. Quatieri*, *R. Dunn*, ``Speaker verification using adapted gaussian mixture models," Digital Signal Processing, vol. 10, no. 1, pp. 19-41, 2000.

4. *A. Davis*, *S. Nordholm*, *R. Togneri*, ``Statistical voice activity detection using low-variance spectrum estimation and an adaptive threshold," IEEE Trans. on Audio, Speech, and Language Processing, vol. 14, no. 2, March 2006.

5. *S. Tranter*, *D. Reynolds*, ``An overview of automatic speaker diarization systems," IEEE trans. on audio, speech and language processing, vol. 14, no. 5, pp. 1557-1565, September 2006.

6. *M. Tuononen*, *R. González Hautamäki*, *P. Fränti*, ``Performance of voice activity detection in speech applications," IEEE Trans. on Information Forensics and Security, 2007, submitted.

7. *M. Sambur*, ``Selection of acoustic features for speaker identification," IEEE trans. on Acoustics, Speech, and Singal Processing, vol. 23, no. 2, pp. 176-182, April 1975.

8. *T. Kinnunen*, *E. Karpov*, *P. Fränti*, ``Real-time speaker identification and verification," IEEE Trans. on Audio, Speech and Language Processing, vol. 14, no. 1, pp. 277-288, January 2006.

9. *J. Saastamoinen*, *E. Karpov*, *V. Hautamäki*, *P. Fränti*, ``Accuracy of MFCC based speaker recognition in series 60 device," J. of Applied Signal Processing, no. 17, pp. 2816-2827, September 2005.

10. *R. Tong*, *B. Ma*, *K. Lee*, *C. You*, *D. Zhou*, *T. Kinnunen*, *H. Sun*, *M. Dong*, *E. Ching*, *H. Li*, ``Fusion of acoustic and tokenization features for speaker recognition," in 5th Intl. Sym. on Chinese Spoken Language Processing (ISCSLP 2006), Singapore, December 2006, pp. 494-505.

11. *R. Tucker*, ``Voice activity detection using a periodicity measure," IEE Proc.-I, vol. 139, no. 4, pp. 377-380, 1992.

12. *P. Fränti*, *T. Kaukoranta*, *O. Nevalainen*, ``On the splitting method for vector quantization codebook generation," Optical Engineering, vol. 36, no. 11, pp. 3043-3051, November 1997.

13. *A. de Cheveigne*, *H. Kawahara*, ``YIN, a fundamental frequency estimator for speech and music," The J. of Acoustical Society of America, vol. 111, no. 4, pp. 1917-1930, April 2002.

14. *M. Turunen*, *J. Hakulinen*, *A. Kainulainen*, ``Evaluation of a spoken dialogue system with usability tests and long-term pilot studies: similarities and difference," in 9th Intl. Conf. on Spoken Language Processing (ICSLP 2006), Pennsylvania, September 2006, pp. 1057-1060.

Dissertations at the Department of Computer Science and Statistics

Rask, Raimo. Automatic Estimation of Software Size during the Requirements Specification Phase - Application of Albrecth's Function Point Analysis Within Structured Methods. Joensuun yliopiston luonnontieteellisiä julkaisuja, 28 - University of Joensuu. Publications in Sciences, 28. 128 p. Joensuu, 1992.

Ahonen, Jarmo. Modeling Physical Domains for Knowledge Based Systems. Joensuun yliopiston luonnontieteellisiä julkaisuja, 33. 127 p. Joensuu, 1995.

Kopponen, Marja. CAI in CS. University of Joensuu, Computer Science, Dissertations 1. 97 p. Joensuu 1997.

Forsell, Martti. Implementation of Instruction-Level and Thread-Level Parallelism in Computers. University of Joensuu, Computer Science, Dissertations 2. 121 p. Joensuu 1997.

Juvaste, Simo. Modeling Parallel Shared Memory Computations. University of Joensuu, Computer Science, Dissertations 3. 190 p. Joensuu 1998.

Ageenko, Eugene. Contex-based Compression of Binary Images. University of Joensuu, Computer Science, Dissertations 4. 111 p. Joensuu 2000.

Tukiainen, Markku. Developing a New Model of Spreadsheet Calculations: A Goals and Plans Approach. University of Joensuu, Computer Science, Dissertations 5. 151 p. Joensuu 2001.

Eriksson-Bique, Stephen. An Algebraic Theory of Multidimensional Arrays. University of Joensuu, Computer Science, Dissertations 6. 278 p. Joensuu 2002.

Kolesnikov, Alexander. Efficient Algorithms for Vectorization and Polygonal Approximation. University of Joensuu, Computer Science, Dissertations 7. 204 p. Joensuu 2003.

Kopylov, Pavel. Processing and Compression of Raster Map Images. University of Joensuu, Computer Science, Dissertations 8. 132 p. Joensuu 2004.

Virmajoki, Olli. Pairwise Nearest Neighbor Method Revisited. University of Joensuu, Computer Science, Dissertations 9. 164 p. Joensuu 2004.

Suhonen, Jarkko. A Formative Development Method for Digital Learning Environments in Sparse Learning Communities. University of Joensuu, Computer Science, Dissertations 10. 154 p. Joensuu 2005.

Xu, Mantao. K-means Based Clustering and Context Quantization. University of Joensuu, Computer Science, Dissertations 11. 162 p. Joensuu 2005.

Kinnunen, Tomi. Optimizing Spectral Feature Based Text-Independent Speaker Recognition. University of Joensuu, Computer Science, Dissertations 12. 156 p. Joensuu 2005.

Kärkkäinen, Ismo. Methods for Fast and Reliable Clustering. University of Joensuu, Computer Science, Dissertations 13. 108 p. Joensuu 2006.

Tedre, Matti. The Development of Computer Science: A Sociocultural Perspective. University of Joensuu, Computer Science, Dissertations 14. 502 p. Joensuu 2006.

Akimov, Alexander. Compression of Digital Maps. University of Joensuu, Computer Science, Dissertations 15. 116 p. Joensuu 2006.

Vesisenaho, Mikko. Developing University-level Introductory ICT Education in Tanzania: A Contextualized Approach. University of Joensuu, Computer Science, Dissertations 16. 199 p. Joensuu 2007.

Huang, Haibin. Lossless Audio Coding for MPEG-4. University of Joensuu, Computer Science and Statistics, Dissertations 17. 86 p. Joensuu 2007.

Mozgovoy, Maxim. Enhancing Computer-Aided Plagiarism Detection. University of Joensuu, Computer Science and Statistics, Dissertations 18. 131 p. Joensuu 2007.

Kakkonen, Tuono. Framework and Resources for Natural Language Parses Evaluation. University of Joensuu, Computer Science and Statistics, Dissertations 19. 131 p. Joensuu 2007.

Podlasov, Alexey. Processing of Map Images for Improving Quality and Compression. University of Joensuu, Computer Science and Statistics, Dissertations 20. 93 p. Joensuu 2007.

Bednarik, Roman. Methods to Analyze Visual Attention Strategies: Applications in the Studies of Programming. University of Joensuu, Computer Science and Statistics, Dissertations 21. 188 p. Joensuu 2007.

Hautamäki, Ville. Improving Pattern Recognition Methods for Speaker Recognition. University of Joensuu, Computer Science and Statistics, Dissertations 22. 126 p. Joensuu 2008.