Feature Selection Based on Genetic Algorithms for Speaker Recognition

Maider Zamalloa, Germán Bordel, Luis Javier Rodríguez, Mikel Peñagarikano

Grupo de Trabajo en Tecnologías del Software
Departamento de Electricidad y Electrónica, Facultad de Ciencia y Tecnología, UPV/EHU
mzamalloa001@ikasle.ehu.es

Abstract

The Mel-Frequency Cepstral Coefficients (MFCC) and their derivatives are commonly used as acoustic features for speaker recognition. The issue arises of whether some of those features are redundant or dependent on other features. Probably, not all of them are equally relevant for speaker recognition. Reduced feature sets allow more robust estimates of the model parameters. Also, less computational resources are required, which is crucial for real-time speaker recognition applications using low-resource devices. In this paper, we use feature weighting as an intermediate step towards feature selection. Genetic algorithms are used to find the optimal set of weights for a 38-dimensional feature set, consisting of 12 MFCC, their first and second derivatives, energy and its first derivative. To evaluate each set of weights, speaker recognition errors are counted over a validation dataset. Speaker models are based on empirical distributions of acoustic labels, obtained through vector quantization. On average, weighting acoustic features yields between 15% and 25% error reduction in speaker recognition tests. Finally, features are sorted according to their weights, and the *K* features with greatest average ranks are retained and evaluated. We conclude that combining feature weighting and feature selection allows to reduce costs without degrading performance.[1]

1. Introduction

Feature extraction is a key issue for efficient speaker recognition. Redundant and harmful information should be removed from speech, retaining only those features relevant to classification. Additionally, a reduced feature set would allow more robust estimates of the model parameters, and less computational resources would be required.

Best features are those that help to discriminate among speakers. An optimal feature should have the following properties: (1) high inter-speaker variation, (2) low intra-speaker variation, (3) easy to measure, (4) robust against mimicry, (5) robust against noise and (6) independent of other features. Unfortunately, no single feature fulfils these requirements. High-level speaker features, such as pronunciation patterns, language use, etc., are robust against noise but require speech recognition to get a sequence of words, and a lot of data to estimate acoustic and language models, which adds an intolerable complexity to the task of speaker recognition. So, low-level acoustic features are the most common choice, because are easy to extract, do not require speech recognition and a small amount of data is enough to estimate good models. Their main inconvenience is that may be easily corrupted by background noise and other distortion sources.

State-of-the-art systems use the same short-term spectrum features (Mel-Frequency Cepstral Coefficients, MFCC) for speech and speaker recognition, because MFCC convey not only the frequency distribution identifying sounds, but also the glottal source and the vocal tract shape and length, which are speaker specific features. Additionally, it has been shown that dynamic information improves significantly the performance of recognizers, so MFCC, energy and their first and second derivatives are commonly used as features. Depending on the acoustic front-end, the resulting feature vectors may have from 20 to 50 components.

Using the voice as a biometric to verify or search for the identity of users is currently the most natural (non-intrusive) way of solving the problem of unauthorized access to critical or private resources (buildings, web accounts, etc.). High security applications require speaker recognition to perform almost perfectly, which is quite difficult, especially when dealing with hundreds of speakers and thousands of possible imposters. However, this issue can be conveniently addressed with computationally powerful back-ends and high volumes of speech data, which allow to estimate robust models even for 40-dimensional feature vectors. Companies adopting this approach can generally afford the cost of computational resources, and their employees/clients use to cooperate in the sometimes tedious phase of speaker enrolment/profiling.

Other applications do not emphasize security but instead automatic adaptation/customization of products and services. Speaker identification is a natural way of customizing many services, by first identifying and then retrieving information about clients. Also, after identification, client choices or activities can be tracked and stored, improving their profiles for further interactions. In these applications, transparency and naturalness is critical. Clients should be aware that their activities are being tracked, their voice recorded and their profile information stored, but these actions should not interfere with the service itself. Depending on the interface, speakers might be continuously tracked, or just identified when the service is started. In any case, they cannot be asked even for just a few seconds of speech data to create accurate profiles, nor interactions delayed due to a computationally expensive search of the client's profile. Moreover, clients may be accessing the service through a portable or embedded device with low storage and computational capabilities. In this case, real-time operation becomes the most critical issue to allow naturalness and transparency.

[1] This work has been partially funded by the Basque Government, under program SAIOTEK, project S-PE04UN18.

For this latter kind of applications, 40-dimensional feature vectors do not seem suitable. A further feature set reduction is needed. The problem of feature extraction and selection is sometimes formulated as a linear transformation which projects feature vectors on a transformed subspace defined by relevant directions. Given a D-dimensional feature vector X, a $K*D$ matrix $\boldsymbol{A}$ is applied to get a K-dimensional vector Y of transformed features ($K<D$). The matrix $\boldsymbol{A}$ is estimated so that, from the point of view of classification, redundancy is removed and relevant information retained. This should, at least, optimize the performance for the target value of K, but it may even outperform the baseline feature set, due to the removal of harmful or confusing features and, more probably, to better (more robust) estimates of model parameters. The following methods have been proposed (among others):

- *Principal Component Analysis* (PCA), an old technique of multivariate statistical analysis [1], consists of computing the eigenvectors of the $D*D$ covariance matrix Σ, then sorting them according to the corresponding eigenvalues, in descending order, and finally building the projection matrix $\boldsymbol{A}$ (called *Karhunen-Loeve Transform*, KLT) with the largest K eigenvectors (i.e. the K directions of greatest variance). Each feature vector X is then pre-processed according to the expression $Y=\boldsymbol{A}(X-\mu)$, where μ represents the mean feature vector. KLT decorrelates the features and provides the smallest possible reconstruction error among all linear transforms, i.e. the smallest possible mean-square error between the data vectors in the original D-feature space and the data vectors in the projection K-feature space. Unfortunately, this does not equal minimizing classification error.
- *Linear Discriminant Analysis* (LDA) [2] attempts to find the transform $\boldsymbol{A}$ that maximizes a criterion of class separability. This is done by computing the within-class and between-class variance matrices, $\boldsymbol{W}$ and $\boldsymbol{B}$, then finding the eigenvectors of $\boldsymbol{W}^{-1}\boldsymbol{B}$, sorting them according to the eigenvalues, in descending order, and finally building the projection matrix $\boldsymbol{A}$ with the largest K eigenvectors (which define the K most discriminative hyperplanes). LDA assumes that all classes share a common within-class covariance, and a single Gaussian distribution per class. LDA also assumes that classes are linearly separable. Additionally, as any supervised approach, it requires labelling samples with speaker identities.
- *Independent Component Analysis* (ICA) [3] is a more recent technique that aims to reduce redundancy in the original feature space. Whereas PCA removes second order dependencies, ICA removes also higher order dependencies, by minimizing the mutual information between the features, thus projecting them on the directions of maximum independence. In fact, ICA was originally designed to solve the problem of blind source separation. Observed signals are assumed to be a linear combination of some unknown statistically independent non-Gaussian source signals. The task of ICA is to recover the source signals from the observed signals, i.e. to find those directions that are best for separating the sources. Once the full $D*D$ matrix is estimated, the K projection vectors with greatest L_2-norms may be retained to build the transformation matrix $\boldsymbol{A}$.

Though linear transforms combine in an elegant way feature extraction and feature selection, these two steps are more usually uncoupled. As noted above, MFCC and their derivatives are a widely accepted representation in the speaker recognition community. However, depending on the application and the available computational resources, this representation must be further reduced. Strictly speaking, feature selection consists of determining an optimal subset of features by exhaustively exploring all the 2^D possible combinations. Most feature selection procedures use the classification error as the evaluation function. This makes exhaustive search computationally infeasible in practice, even for moderate values of D. The simplest method consists of evaluating the D features individually and selecting the K most discriminative features, but it does not take into account dependencies among features. So a number of suboptimal heuristic search techniques have been proposed in the literature, which essentially trade-off the optimality of the selected subset for computational efficiency [4].

Genetic Algorithms (GAs for short), introduced by Holland in 1975 [5], are randomized heuristic search techniques based on biological evolution strategies, with three basic operations: selection of the fittest, crossover and mutation. GAs are usually applied in complex optimization problems. Candidate solutions are represented by individuals (or chromosomes) in a large population. Initial solutions may be randomly generated or obtained by other means. Then GAs iteratively drive the population to an optimal point according to a complex metric (called fitness or evaluation function) that measures the performance of the individuals in a target task. The fittest individuals are selected and their chromosomes mixed, mutated or taken unchanged to the next generation. A major advantage of the GAs over other heuristic search techniques is that they do not rely on any assumption about the properties of the evaluation function. Multiobjective evaluation functions (e.g. combining the accuracy and the cost of classification) can be defined and used in a natural way [6] [7]. GAs can easily encode decisions about selecting or not selecting features as sequences of boolean values, allow to smartly explore the feature space by retaining those decisions that benefit the classification task, and simultaneously avoid local optima due to their intrinsic randomness.

GAs have been recently applied to feature extraction and selection in speaker recognition tasks [8] [9] [10]. Additionally, GAs can be easily generalized to feature weighting, by encoding not boolean decisions but feature weights, which is shown to improve recognition performance [8]. In this work, a genetic algorithm is used to find the optimal set of weights for a 38-dimensional feature set, consisting of 12 MFCC, their first and second derivatives, energy and its derivative. Weights are encoded as 8-bit integers, so each individual (representing a set of weights) is encoded by 304 bits. A speaker recognition benchmark is used to evaluate the fitness of individuals. The fittest individual in the last generation represents the optimal set of weights. Naive speaker models are used, based on empirical distributions of acoustic labels. A database of read speech in Spanish, including 204 speakers, is used for the experiments. Finally, feature selection is implemented by retaining the K best ranked features according to their weights.

The rest of the paper is organized as follows. The speaker recognition system and the GA-based search of the optimal weights are described in Section 2. The experimental setup is described in Section 3, including the speech database used to train and test speaker models and the tuning phase of the GA. Two series of speaker recognition experiments are presented and discussed in Section 4: (1) using feature weights and (2) selecting a varying number of features. Finally, Section 5 summarizes our approach and outlines some methodological improvements we are currently working on.

2. Methodology

2.1. The speaker recognition system

2.1.1. Acoustic front-end

Speech, acquired at 16 kHz, is analysed in frames of 25 milliseconds (400 samples), at intervals of 10 milliseconds. A Hamming window is applied and a 512-point FFT computed. The FFT amplitudes are then averaged in 24 overlapped triangular filters, with central frequencies and bandwidths defined according to the Mel scale. A Discrete Cosine Transform is finally applied to the logarithm of the filter amplitudes, obtaining 12 Mel-Frequency Cepstral Coefficients (MFCC). To increase robustness against channel distortion, Cepstral Mean Normalization (CMN) [11] is applied on an utterance-by-utterance basis. The first and second derivatives of the MFCC, the frame energy (E) and its derivative are also computed, thus yielding a 38-dimensional feature vector.

2.1.2. Vector Quantization

Vector Quantization (VQ) [12] is applied to search for an optimal codebook of $L=256$ centroids which minimizes the average distortion (Euclidean distance) in quantifying feature vectors in the training set (which includes all the speakers). Then, each feature vector in the database is replaced by the index of the closest centroid. The same procedure is applied during recognition. Each input utterance is analysed to get a sequence of feature vectors $X=\{X(1),X(2),...,X(T)\}$, and each $X(i)$ is assigned the index of the nearest centroid in the codebook, $Y(i)$, obtaining a sequence of acoustic labels $Y=\{Y(1),Y(2),...,Y(T)\}$.

2.1.3. Speaker models

The classical VQ approach to speaker recognition consists of training a specific codebook for each speaker. Then, input utterances are classified by choosing the codebook for which the accumulated distortion in quantifying acoustic vectors is minimum. In this work, speaker models are not VQ codebooks but distributions of VQ labels. A single codebook is computed and shared by all the speakers, and the frequencies of VQ labels stored as parameters. These simple models have been successfully used for speaker adaptation through speaker clustering in speech recognition tasks [13]. Let $U(i)$ be the training subset corresponding to speaker i, $c(i)$ the number of VQ labels in $U(i)$, and $c(k,i)$ the number of times the label k appears in $U(i)$. Then, the conditional probability $P(k|i)$ can be empirically estimated as follows:

$$P(k \mid i) = \frac{c(k,i)}{c(i)} . \tag{1}$$

Finally, assuming that successive labels are independent, the conditional probability of a sequence of labels $Y=\{Y(1),Y(2), ...,Y(T)\}$, given speaker i, can be computed as follows:

$$P(Y \mid i) = \sum_{t=1}^{T} P(Y(t) \mid i) . \tag{2}$$

2.1.4. Speaker recognition

Assuming that input utterances are produced by S known speakers, given the sequence of labels $Y=\{Y(1),Y(2),...,Y(T)\}$, the most likely speaker is selected:

$$\hat{i}(Y) = \arg\max_{i=1,...,S} \{P(i \mid Y)\} . \tag{3}$$

Applying the Bayes rule, it follows:

$$\hat{i}(Y) = \arg\max_{i=1,...,S} \left\{ \frac{P(Y \mid i) P(i)}{P(Y)} \right\} = \arg\max_{i=1,...,S} \{P(Y \mid i) P(i)\} , \tag{4}$$

because maximizing over the set of speakers does not depend on the acoustic sequence. Then, assuming that all speakers have equal *a priori* probabilities, it follows:

$$\hat{i}(Y) = \arg\max_{i=1,...,S} \{P(Y \mid i)\} , \tag{5}$$

and introducing (2) into (5):

$$\begin{aligned} \hat{i}(Y) &= \arg\max_{i=1,...,S} \left\{ \prod_{t=1}^{T} P(Y(t) \mid i) \right\} \\ &= \arg\max_{i=1,...,S} \left\{ \log \prod_{t=1}^{T} P(Y(t) \mid i) \right\} . \\ &= \arg\max_{i=1,...,S} \left\{ \sum_{t=1}^{T} \log P(Y(t) \mid i) \right\} \end{aligned} \tag{6}$$

So, the computational cost of speaker recognition is linear with the number of speakers (S) and with the length of the input utterance (T), involving $S \times T$ memory accesses, $S \times (T-1)$ sums and $S-1$ comparisons. For convenience, it is assumed that label probabilities are stored in logarithmic form.

2.1.5. How do feature weighting and feature selection affect speaker recognition?

Feature weighting affects speaker models only through the VQ process. For each candidate set of feature weights $W=\{w_1,w_2,...,w_D\}$, a different codebook $C(W)$ is computed by first weighting feature vectors in the training set and then using the Euclidean distance to measure distortion in the weighted space. So, $C(W)$ consists of the L centroids that minimize VQ distortion in the weighted space. Obviously, labelling the database also depends on weighting: W is applied to each feature vector X, and the resulting vector $X'=(w_1x_1,w_2x_2,...,w_Dx_D)$ is replaced by the index k corresponding to the closest centroid in $C(W)$.

Feature selection takes place after the optimal weights are found. It consists of retaining the K most relevant features, so that a K-dimensional subspace is extracted from the original D-dimensional feature space. The VQ codebook $C(W,K)$ is computed in the weighted K-dimensional subspace, and the database is labelled by first picking the K most relevant components of each feature vector, then applying the corresponding weights and finally replacing the resulting vector by the index of the closest centroid in *C(W,K)*.

2.2. GA-based search for the optimal weights

The well-known *Simple Genetic Algorithm* (SGA) [14] is employed to search for the optimal set of weights. As noted above, for each candidate set of weights $W=\{w_1,w_2,...,w_D\}$, a codebook *C(W)* is computed and the whole database labelled according to *W* and *C(W)*. Then, *c(k,i)* and *c(i)* are counted for each training subset *U(i)*, and the speaker models $M=\{P(k|i)|k\in[1..L],i\in[1..S]\}$ are estimated using (1). Finally, utterances in the validation set $V=\{V(1),V(2),...,V(S)\}$ are classified, and the classification accuracy used as fitness function:

$$F(W)=\sum_{i=1}^{S}\sum_{\forall Y\in V(i)}\delta\left(\hat{i}(Y)=i\right), \qquad (7)$$

where:

$$\delta(x)=\begin{cases}1 & if\ x=true\\ 0 & if\ x=false\end{cases}. \qquad (8)$$

Once all the candidates are evaluated, some of them (usually the fittest ones) are selected, mixed and mutated in order to get the population for the next generation. SGA allows a number of individuals (the fittest ones) to survive for the next generation, which is called *elitism*. The simplest case of elitism, which consists of keeping the fittest individual, is applied. This guarantees that the fitness of the fittest individual increases monotonically with successive generations. If that increase is smaller than a given threshold, or a maximum number of generations is reached, the algorithm stops and the fittest individual (i.e. the set of weights yielding the highest classification accuracy) is returned. Finally, a third independent set of utterances is used to test the optimal set of weights.

The convergence properties of the SGA depend on several parameters:

- *Representation.* Only relevant features should be encoded, each allocated by a suitable number of bits. If too many bits are allocated, some of them will be redundant (i.e. their value will not affect fitness), computational resources will be spent in exploring an oversized feature space and convergence will be slowed down. Inversely, if not enough bits are allocated, the algorithm will not be able to reach the best solution.
- *Population size.* It must be large enough to ensure diversity (i.e. capacity to evolve), which allows the algorithm to avoid local optima and move towards the best solution. But a large population causes the algorithm to converge slowly. So a balance must be found between the ability to evolve and the speed of convergence. In practice, population size must be proportional to the size of the representation.
- *Initial population.* The first generation of individuals can be randomly generated by the SGA, or obtained by other means (for instance, by applying some simple initialisation engine or some heuristic or prototypical knowledge).
- *Selection procedure.* Once all the individuals are evaluated, pairs of them are selected to breed the next generation. The most common approaches are *fitness proportional selection* (also known as *roulette-wheel selection*), *rank selection* and *tournament selection*. In fitness proportional selection the probability of picking an individual is proportional to its fitness value. In rank selection, the individuals are sorted by fitness, and the probability of picking one of them is proportional to its rank. Finally, in tournament selection some individuals are randomly chosen and the fittest among them is picked.
- *Crossover type.* The crossover operator recombines the genes of two individuals to create two offspring. This is typically accomplished by defining *n* breaking points and dividing the original chromosomes into *n+1* segments. Even segments are then interchanged to get the offspring. The number of breaking points determines the type of crossover: one-point crossover, two-point crossover, etc.
- *Mutation rate.* Mutation consists of inverting the value of a few random bits in the chromosome. It is used to introduce small variations that help decreasing the chances of getting to local optima. The mutation rate determines the probability that mutations take place after crossover. It is usually fixed to very low values, 0.05 or less.

2.3. Feature selection

Though feature weighting should probably improve the performance of the speaker recognition system, this work is focused on reducing storage and computational costs by selecting the *K* most relevant features. Our approach is based on the hypothesis that weights somehow measure the contribution of features to the performance of the recognizer. In other words, the more discriminative a feature is the higher its weight will be. Under this assumption we can easily define a reduced set of features by selecting those best ranked according to their weights.

3. Experimental setup

3.1. The speech database

A phonetically balanced database in Spanish, called *Albayzín* [15], is used for the experiments. *Albayzín,* recorded at 16 kHz in laboratory conditions, was originally designed to train acoustic models for speech recognition. It contains 204 speakers, each speaker contributing at least 25 utterances and each utterance lasting 3.55 seconds on average.

Using the classification accuracy to evaluate feature weights makes the optimization process very costly. To speed up the evaluation of the proposed methodology, instead of using the whole database, a partition Π_1 is defined which consists of 10 partially overlapped subsets of 20 speakers, involving 164 speakers all together. Each subset is further divided into three independent datasets: *training* (5 utterances per speaker), *validation* (10 utterances per speaker) and *test* (10 utterances per speaker). The training set is used to compute the VQ codebook and to estimate speaker models. The validation set is used by the GA to search for the optimal feature weights. Finally, the test set is used to evaluate the performance of weighted features in speaker recognition experiments.

As will be shown below, average feature ranks and weights, computed over the 10 speaker subsets of Π_1, are used, respectively, to select and weight the *K* most relevant features, for several values of *K*. To evaluate each set of *K* features a new partition Π_2 is defined, consisting of 12 partially overlapped subsets of 20 speakers involving all the speakers in the database. Two independent corpora are defined for each subset of speakers: *training* (5 utterances per speaker) and *test* (10 utterances per speaker). VQ codebooks and speaker models are estimated using samples in the training corpus and speaker recognition experiments are carried out over the test corpus.

3.2. Tuning the GA

The SGA is implemented by using ECJ, a *Java-based Evolutionary Computation and Genetic Programming Research System*, presently developed at George Mason University's Evolutionary Computation Laboratory and released under a special open source license [16]. ECJ shows very interesting features, including a flexible breeding architecture, arbitrary representations, fixed and variable length genomes, several multiobjective optimization methods and many selection operators.

Preliminary experimentation has been carried out to adjust the parameters that control the performance and the convergence of the SGA. It has been observed that populations of 50 individuals need at most 30 generations to converge, so no convergence criterion is applied and a fixed number of 30 generations is established. Chromosomes consist of 38 genes, each encoding a feature weight. To reduce computational costs as much as possible, 8 bits have been allocated for each weight. So, allowed gene values range from 0 to 255. Offspring is bred by first selecting and then mixing two parents in the current population. One of the parents is selected according to the fitness-proportional criterion. The second is selected according to the tournament method, by picking the fittest of 7 randomly chosen individuals (the choice of 7 has proven good in the experiments and is also suggested by the manufacturers of the toolkit). This mixed approach seems suitable because fitness-proportional selection guarantees that the fittest individuals are picked, and tournament selection introduces diversity, which is good for avoiding local optima. Crossover type and mutation probability have been established by picking those values yielding the best speaker recognition results. Finally, as noted above, the simplest case of elitism is applied by keeping the fittest individual for the next generation. Tuned values are shown in Table 1.

4. Experimental results

4.1. Feature weighting

Two series of experiments were carried out by using 2 and 3 training utterances per speaker over Π_1. Hereafter we will refer to them as the 2U and 3U experiments, respectively. GA-based optimization was applied, using the settings shown in Table 1, to get the optimal set of feature weights for each configuration.

Table 2 shows the average performance obtained using non-weighted and weighted features for the test and validation corpora in the 2U and 3U experiments. Error rates over the test corpora using weighted features were, on average, 2.95 and 1.05 points lower than those achieved using non-weighted features, which means error reductions of 24.58% and 14.68%, respectively. As may be expected, error rates over the validation corpora were significantly lower, since feature weights were searched specifically to maximize the performance over them. In the 2U experiments, recognition rates using weighted features were, on average, 7.5 points better than those achieved using non-weighted features, which means a 61.22% error reduction. The average error reduction was even larger in the 3U experiments (70.63%, 4.45 points). These results provide evidence that further improvements in speaker recognition performance can be attained by weighting acoustic features. They also validate the use of GAs to search for an optimal set of feature weights.

Table 1. Tuned settings of the SGA parameters.

Parameter			Setting
Population size			50
Number of generations			30
Chromosome size (number of genes)			38
Gene Values	Minimum		0
	Maximum		255
Genetic operations	Crossover	Type	Two-point
		Rate	0.8
	Mutation	Rate	0.05
Selection	First Parent		Fitness-Proportional
	Second Parent		Tournament (Size: 7)
Elitism			1

Table 2. Average speaker recognition error rates over 10 different sets of 20 speakers, using non-weighted and weighted features for the 2U/3U experiments (2/3 utterances for training, 10 utterances for validation and 10 utterances for test).

	Test set		Validation set	
	Non-Weighted	Weighted	Non-Weighted	Weighted
2U	12.00	9.05	12.25	4.75
3U	7.15	6.10	6.30	1.85

Table 3. Average feature weights, computed over 20 sets of GA-based optimal weights, corresponding to the 2U/3U experiments.

Feature	Weight	Feature	Weight	Feature	Weight	Feature	Weight
c01	103	d01	106	dd01	158	E	89
c02	170	d02	151	dd02	82	dE	137
c03	142	d03	63	dd03	178		
c04	155	d04	69	dd04	199		
c05	123	d05	94	dd05	103		
c06	177	d06	100	dd06	152		
c07	186	d07	200	dd07	177		
c08	215	d08	133	dd08	106		
c09	234	d09	197	dd09	97		
c10	231	d10	111	dd10	117		
c11	173	d11	169	dd11	185		
c12	146	d12	118	dd12	164		

Finally, a single set of weights was computed by averaging the optimal weights obtained over Π_1 in the 2U and 3U experiments. As shown in Table 3, the MFCC c08, c09, c10 and the MFCC first derivative d07 seem to be the most relevant features, with average weights greater or equal than 200. Additionally, 15 features have average weights in the range [150,199]: 5 MFCC, 3 MFCC first derivatives and 7 MFCC second derivatives. The least relevant features seem to be the MFCC first derivatives d03, d04 and d05, the MFCC second derivatives dd02 and dd09, and energy (E), with average weights lower than 100. Of course, if we ran the GA-based optimization for all the 164 speakers involved in Π_1, a different set of weights would be probably obtained. However, taking into account that 10 different validation sets and two models per validation set have been averaged, these weights are fair indicators of the relative importance of each feature for speaker recognition, and will be considered hereafter as globally optimal weights.

4.2. Feature selection

To select a subset of K features we rank them according to their relative importance for speaker recognition. Assuming that the assigned weights are fair estimates of the importance of features, a straightforward procedure could be defined based on the average weights shown in Table 3. First, features would be sorted according to their average weights, and then the K best ranked features would be selected. However, optimal weights depend highly on the speaker models and the validation dataset used by the GA. Average weights over the 2U and 3U experiments show a high variability (i.e. high standard deviations), since 20 different configurations are considered. So, average weights are not a good choice to rank features. They only approximate the weight values but not their ranks. In other words, it does not seem suitable in this case to define a feature rank as the rank of the average feature weight. A feature rank can be also defined as the average of the optimal weight ranks in the 2U and 3U experiments. This appears to be a more suitable approach, since optimal weight ranks include accurate information about the relative importance of features. Summarizing, for selecting features we will take into account not the values but the ranks of optimal weights.

Table 4 shows the global and partial average ranks computed over the 2U and 3U experiments. Besides the feature name, each row includes the average rank computed over a set of experiments: 2U+3U (global ranking), 2U and 3U. Partial rankings do not match exactly, but most features appear reasonably close in both columns. If the global ranking based on average ranks and the ranking that can be obtained from average weights (Table 3) are compared, some similarities can be observed. Among the 10 best ranked features, 8 are present in both rankings: c06, c07, c08, c09, c10, d07, d09 and dd04. It is interesting to note that 5 of them are medium/high-index MFCC. The three best ranked features are in both cases c08, c09 and c10, though ranked in different order: c09, c10 and c08 attending to the average weights, and c09, c08 and c10 attending to the average ranks. Additionally, it must be noted that c08, c09 and c10 are consistently ranked as the best features, because their optimal weights are among those with the lowest standard deviations. Finally, in both cases energy (E) is ranked as one of the worst features. Its derivative (dE) is better ranked: 22 attending to the average weight, and 11 attending to the average rank. This latter result is somehow surprising, but can be explained by examining the partial ranks corresponding to experiments 2U and 3U in Table 4: dE is assigned rank 3 in the 2U experiments and rank 29 in the 3U experiments. So, dE seems to be useful only when few enrolment data are available to estimate speaker models.

Table 4. Average feature ranks for the 2U+3U, 2U and 3U experiments.

	2U+3U		2U		3U	
Rank	Feature	Average Rank	Feature	Average Rank	Feature	Average Rank
1	***c09***	3.97	*c09*	3.38	*c09*	4.57
2	***c08***	8.79	*c08*	10.00	*c08*	7.57
3	***c10***	11.40	*dE*	10.88	*c10*	11.29
4	***c07***	13.60	*c10*	11.50	*d07*	11.29
5	***dd04***	15.00	*c07*	14.25	*c07*	12.86
6	***d07***	15.50	*d09*	15.00	*dd04*	13.71
7	***c06***	15.70	*c11*	15.75	*d11*	14.14
8	***dd07***	15.80	*c06*	15.88	*dd07*	14.86
9	***c02***	16.10	*c02*	16.13	*c06*	15.43
10	***d09***	16.10	*dd03*	16.25	*c02*	16.00
11	***dE***	16.90	*dd04*	16.38	*d02*	16.00
12	***dd03***	17.10	*dd07*	16.75	*d09*	17.29
13	***c11***	17.30	*dd11*	17.25	*dd11*	17.43
14	***dd11***	17.30	*d06*	17.88	*dd03*	17.86
15	***d02***	17.60	*dd12*	17.88	*c11*	18.86
16	***dd12***	18.40	*dd01*	18.38	*c04*	19.00
17	***d11***	18.80	*d02*	19.25	*dd12*	19.00
18	***dd01***	19.90	*d07*	19.63	*c12*	19.29
19	***c12***	20.10	*dd06*	19.88	*d12*	19.43
20	***c04***	20.20	*c03*	20.13	*dd09*	20.57
21	***c03***	20.60	*d08*	20.38	*c03*	21.14
22	***dd06***	20.70	*c12*	21.00	*d08*	21.14
23	***d08***	20.80	*dd08*	21.00	*c05*	21.29
24	***d01***	21.30	*dd05*	21.25	*d01*	21.29
25	***d10***	22.40	*c04*	21.38	*dd01*	21.43
26	***dd08***	22.90	*d01*	21.38	*dd06*	21.43
27	***c05***	23.10	*d10*	21.75	*c01*	21.71
28	***d06***	23.30	*dd10*	23.00	*d05*	21.71
29	***dd10***	23.40	*d11*	23.50	*dE*	22.86
30	***dd09***	23.50	*dd02*	23.75	*d10*	23.00
31	***d12***	24.20	*E*	24.00	*dd10*	23.86
32	***c01***	24.50	*d04*	24.38	*dd08*	24.86
33	***dd02***	25.00	*c05*	24.88	*dd02*	26.29
34	***dd05***	25.30	*d03*	25.63	*E*	27.14
35	***d05***	25.30	*dd09*	26.38	*d04*	28.00
36	***E***	25.60	*c01*	27.25	*d06*	28.71
37	***d04***	26.20	*d05*	28.88	*dd05*	29.29
38	***d03***	27.50	*d12*	28.88	*d03*	29.43

4.3. Evaluating reduced sets of features

Based on the global ranking shown in Table 4, speaker recognition experiments were run over Π_2, using reduced sets of K weighted and non-weighted features. First, for each subset of speakers in Π_2, specific VQ codebooks and speaker models were estimated, by using R training utterances per speaker (R = 2, 3, 4 and 5). Then, for each set of K weighted and non-weighted features, for each training size R and for each subset of speakers, recognition experiments were run over the corresponding test corpus. Finally, the recognition accuracy was averaged over the 12 subsets of speakers to get a global and more reliable performance measure. Initially, experiments were carried out for *K* = 35, 30, 25, 15, 10 and 5. However, after analysing the results, we found it convenient to get more resolution for lower values of *K* (between 5 and 10). So, four new experiments were carried out for *K* = 6, 7, 8, and 9. Table 5 shows the average error rates obtained in those experiments, using *K* weighted and non-weighted features and 4 levels of training (from 2 to 5 utterances per speaker).

Table 5. Error rates in speaker recognition experiments for reduced sets of weighted and non-weighted features, using 4 levels of training (2-5 utterances per speaker). Results with the full 38-dimensional feature vectors are shown too for reference.

	Number of features										
Training	38	30	25	20	15	10	9	8	7	6	5
	Non-weighted										
2	10.54	14.42	15.58	17.21	20.88	21.29	22.33	29.88	31.71	41.33	41.75
3	6.96	8.75	10.46	11.42	13.96	13.92	14.04	21.29	22.33	32.58	31.13
4	5.00	6.63	8.04	8.92	11.54	11.88	11.67	18.38	18.29	26.88	25.67
5	4.33	4.38	6.50	6.17	8.08	8.17	8.67	15.04	14.79	22.42	21.88
	Weighted										
2	8.88	10.83	13.71	15.21	19.04	20.38	19.42	28.08	28.38	41.79	42.25
3	5.71	6.96	8.00	8.71	12.67	11.67	12.13	19.92	20.00	31.92	29.96
4	3.63	4.71	5.83	7.00	9.21	8.79	9.88	16.13	15.75	24.96	24.92
5	2.83	3.33	3.92	4.79	6.83	7.04	6.96	12.88	13.00	22.71	22.08

As shown in Figure 1, speaker recognition performance degraded at a small pace and kind of linearly from *K*=38 to *K*=10. Lower values of *K* led to large degradations, suggesting that the 10 best ranked features contain the most relevant information about speaker identity. A more detailed inspection reveals that performance degraded monotonically as the number of features was reduced from 38 to 15. Then, from 15 to 10 features the performance improved slightly in some cases (in particular, when using 3 and 4 utterances per speaker), suggesting that the feature ranking could be further adjusted in that range. Finally, from 10 to 5 features the performance degraded fast and irregularly, suggesting the lack of relevant features and possibly a higher sensitivity to rank disorder. For instance, moving from 9 to 8 features (i.e. discarding c02) produced a large degradation, whereas moving from 8 to 7 features (i.e. discarding dd07) did hardly affect performance.

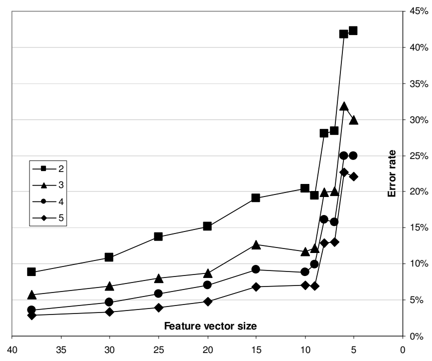

Figure 1. Error rates increase as the feature vector is reduced. A sharp change in slope takes place for K=10.

Finally, after verifying that feature weighting can be suitably applied to improve speaker recognition performance, and combined with feature selection to reduce storage and computational costs, Table 5 reveals that performance can be also improved by increasing the robustness of speaker models (i.e. by increasing the number of training utterances per speaker). For instance, with 10 weighted features, the error rate reduces from 20% when using 2 training utterances, to 7% when using 5 training utterances. This means that a suitable balance can always be found between the number of training utterances and the number of features. In this case, using 10 weighted features with 4 training utterances per speaker yields the same performance as using 38 weighted features with 2 training utterances per speaker. In other words, increasing the robustness of speaker models allows to reduce storage and computational costs without degrading performance, as can be graphically seen in Figure 2.

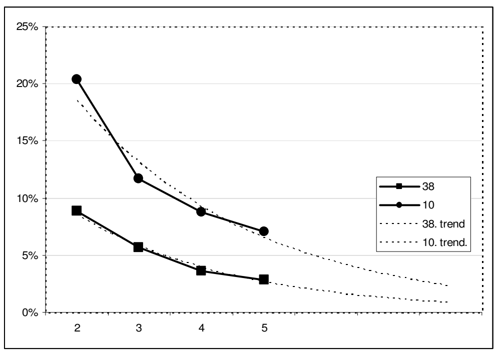

Figure 2. Error rates in speaker recognition experiments using 38 and 10 weighted features, as a function of the number of utterances used to estimate speaker models. Graphs are fitted to quadratic functions that show trends in both cases.

5. Conclusions and future work

For speaker recognition applications requiring real-time operation or running on low-resource devices, reducing the number of features is crucial, since storage and computational costs may be lightened. Additionally, model parameters may be estimated more robustly. GAs can be suitably applied to search for the most relevant features, or more generally, to search for an optimal set of feature weights.

In this work, we followed a hybrid approach, by first obtaining the optimal weights and then using them to cut the representation. Empirical distributions of VQ labels were used as speaker models. A phonetically balanced database in Spanish, containing 204 speakers, was used as benchmark.

First, the potential benefit of weighting acoustic features for speaker recognition was evaluated over a partition Π_1 consisting of 10 subsets of 20 speakers. Two series of experiments were carried out by using 2 and 3 training utterances per speaker to estimate VQ codebooks and speaker models. A genetic algorithm was applied to search for the weights minimizing speaker recognition errors over validation datasets. It was found that using weighted features reduced error rates by 20%, on average, in speaker recognition experiments over independent test data. Finally, a single set of feature weights was computed by averaging the optimal weights obtained for the 10 subsets of speakers in the two series of experiments.

A feature selection procedure was designed based on the average ranks of features. Features were sorted in descending order according to the weights obtained in the previous experiments. Then the *K* features with greatest average ranks were selected. Speaker recognition experiments were run over a second partition Π_2, consisting of 12 subsets of 20 speakers, to investigate how the accuracy was affected as the feature vector was reduced, by selecting the *K* most relevant features, for *K=30, 25, 20, 15, 10, 9, 8, 7, 6* and *5*. Four series of experiments were carried out by using 2, 3, 4 and 5 training utterances per speaker. It was found that reducing *K* degraded performance in most cases. Average error rates grew *slowly* from *K=30* to *K=10* and *rapidly* from *K=10* to *K=5*. This may indicate that the 10 best ranked features contain the most relevant information about speaker identity. On the other hand, as for Π_1, weighting features yielded significant error reductions. Also, speaker models provided better performance as the size of the training dataset increased. For instance, with 10 weighted features, the error rate reduced from 20% when using 2 training utterances per speaker, to 7% when using 5 training utterances per speaker. Finally, important savings in storage and computational costs can be attained by combining feature selection and feature weighting.

Future work includes two methodological improvements to the work presented in this paper:

- *Finding optimal weights for each K*. In this paper, an optimal set of feature weights is found for a 38-dimensional feature space. So, weights are optimal only when using the full representation. If the same procedure was applied for a subset of *K* features, it would very probably lead to a different set of weights. If feature weights are only the means to select the K most relevant features and *K* non-weighted features are eventually used, the proposed methodology is a good alternative. But applying the optimal weights obtained for a 38-dimensional feature space to a reduced subset of *K* features is just a suboptimal approximation. So, finding the optimal set of weights specifically for each *K*-dimensional feature subspace should lead to further improvements.

- *Extending feature weighting to feature transformation*. Weighting can be seen as a special case of linear transformation for which the transformation matrix is diagonal. So, feature weighting can be generalized to feature transformation in a straightforward way. As noted above, such a transformation can be estimated according to different criteria: least mean-square reconstruction error (PCA), maximum class separability (LDA), maximum independence (ICA), etc. But none of those approaches guarantees that classification error is minimized. Genetic algorithms would search, instead, for the transformation that minimizes classification error over validation data, projecting the original features onto an optimal *K*-dimensional subspace.

6. References

[1] I.T. Jolliffe. *"Principal Component Analysis (Second Edition)"*, Springer, 2002.

[2] R.O. Duda, P.E. Hart, D.G. Stork. *"Pattern Classification (Second Edition)"*, Wiley Interscience, 2000.

[3] E. Oja, A. Hyvarinen, J. Karhunen. *"Independent Component Analysis"*, John Wiley & Sons, 2001.

[4] A.K. Jain, R.P.W. Duin, J. Mao. *"Statistical Pattern Recognition: A Review"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 1, pp. 4—37, January 2000.

[5] J.H. Holland. *"Adaptation in natural and artificial systems"*. University of Michigan Press, 1975 (reprinted in 1992 by MIT Press, Cambridge, MA).

[6] J. Yang, V. Honavar. *"Feature subset selection using a genetic algorithm"*, IEEE Intelligent Systems, Vol. 13, No. 2, pp. 44—49, March 1998.

[7] L.S. Oliveira, R. Sabourin, F. Bortolozzi, C.Y. Suen. *"A Methodology for Feature Selection Using Multiobjective Genetic Algorithms for Handwritten Digit String Recognition"*, International Journal of Pattern Recognition and Artificial Intelligence, Vol. 17, No. 6, pp. 903—929, 2003.

[8] D. Charlet, D. Jouvet. *"Optimizing feature set for speaker verification"*, Pattern Recognition Letters, Vol. 18, No. 9, pp. 873—879, September 1997.

[9] M. Demirekler, A. Haydar. *"Feature Selection Using a Genetics-Based Algorithm and its Application to Speaker Identification"*, Proceedings of the IEEE ICASSP'99, pp. 329—332, Phoenix, Arizona, 1999.

[10] C. Charbuillet, B. Gas, M. Chetouani, J.L. Zarader. *"Filter Bank Design for Speaker Diarization Based on Genetic Algorithms"* to appear in Proceedings of the IEEE ICASSP'06, Toulouse, France, May 2006.

[11] A.E. Rosenberg, C.H. Lee, F.K. Soong. *"Cepstral Channel Normalization Techniques for HMM-Based Speaker Verification"*, Proceedings of the ICSLP'94, pp. 1835—1838, Yokohama, Japan, 1994.

[12] Y. Linde, A. Buzo, R.M. Gray. "An Algorithm for Vector Quantizer Design", IEEE Transactions on Communications, Vol. 28, No. 1, pp. 84—95, January 1980.

[13] L.J. Rodríguez, M.I. Torres. *"A Speaker Clustering Algorithm for Fast Speaker Adaptation in Continuous Speech Recognition"*, in P. Sojka, I. Kopecek and K. Pala Eds., Proceedings of the 7th International Conference on Text, Speech and Dialogue (Brno, Czech Republic, September 2004), pp. 433—440, LNCS/LNAI 3206, Springer-Verlag, 2004.

[14] D.E. Goldberg. *"Genetic Algorithms in Search, Optimization and Machine Learning"*, Addison-Wesley, 1989.

[15] F. Casacuberta, R. García, J. Llisterri, C. Nadeu, J.M. Pardo, A. Rubio. *"Development of Spanish Corpora for Speech Research (Albayzín)"*, in G. Castagneri Ed., Proceedings of the Workshop on International Cooperation and Standardization of Speech Databases and Speech I/O Assessment Methods, Chiavari, Italy, 26-28 September 1991, pp. 26—28.

[16] ECJ 13, http://cs.gmu.edu/~eclab/projects/ecj/.