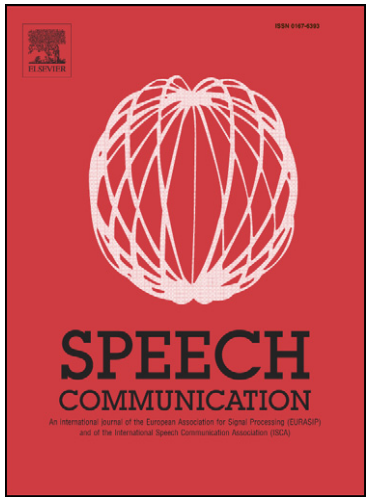
Multi-Style Classification of Speech Under Stress Using Feature Subset Selection Based on Genetic Algorithms

Salvatore Casale, Alessandra Russo, Salvatore Serrano



Multi-Style Classification of Speech Under Stress Using Feature Subset Selection Based on Genetic Algorithms

Salvatore Casale, Alessandra Russo, Salvatore Serrano

PII: S0167-6393(07)00083-0
DOI: 10.1016/j.specom.2007.04.012
Reference: SPECOM 1639

To appear in: *Speech Communication*

Received Date: 1 April 2006
Revised Date: 13 February 2007
Accepted Date: 23 April 2007

Please cite this article as: Casale, S., Russo, A., Serrano, S., Multi-Style Classification of Speech Under Stress Using Feature Subset Selection Based on Genetic Algorithms, *Speech Communication* (2007), doi: 10.1016/j.specom.2007.04.012

Multi-Style Classification of Speech Under Stress Using Feature Subset Selection Based on Genetic Algorithms

Salvatore Casale, Alessandra Russo, Salvatore Serrano

Dipartimento di Ingegneria Informatica e delle Telecomunicazioni
University of Catania
Viale A. Doria, 6 - 95125 Catania, Italy

Abstract

The determination of an emotional state through speech increases the amount of information associated with a speaker. It is therefore important to be able to detect and identify a speaker's emotional state or state of stress. Various techniques are used in the literature to classify emotional/stressed states on the basis of speech, often using different speech feature vectors at the same time. This study proposes a new feature vector that will allow better classification of emotional/stressed states. The components of the feature vector are obtained from a feature subset selection procedure based on Genetic Algorithms. A good discrimination between neutral, angry, loud and Lombard states for the simulated domain of the Speech Under Simulated and Actual Stress (SUSAS) database and between neutral and stressed states for the actual domain of the SUSAS database is obtained.

1 Introduction

Technological progress has allowed an increasing degree of human-machine interaction. This interaction can be improved and accelerated by means of spoken communication. In human-human speech-based communications, emotions play an important role, sometimes playing an even bigger role than the logical information also included in the speech [1]. One important research challenge in the last few years has thus been automatic recognition of the emotional state of a speaker through speech; knowledge of this state can have a series of applications, such as [1]:

- automatic answering machines that can adapt their voice tone to that of the speaker;

- speech recognition systems that can correctly interpret the meaning of words spoken in an ironic or sarcastic manner;
- systems which are able to detect the emotional state of an interlocutor (for clinical diagnosis purposes, for example);
- synthesizers that can generate speech giving the sensation of a particular emotion;
- automatic tutoring systems that can establish a learner's degree of boredom, irritation or intimidation;
- systems that can prevent speakers in a particularly altered emotional state from interacting with automatic recognition systems;
- systems to generate alarms on the basis of the different emotional states of people being monitored;
- entertainment systems and games that can determine a user's emotional state.

The determination of an emotional state is therefore an important task from several points of view. What is studied more frequently than the recognition of the speaker's emotional state in general is classification of speech under stress (e.g., [2][3][4][5]). Several elements introduce stressed voice tones in the production of speech, for example background noise, emergencies, high workloads, strong emotional excitement, etc. It is well-known that the performance of speech recognition algorithms is greatly influenced by the stressful conditions in which speech is produced. Workload task stress significantly impacts recognition performance. Effects of different stressful conditions on speech recognition and efforts to improve the performance of speech recognition algorithms under stressful conditions can be found in literature. Stress classification cannot only be used to improve the robustness of speech recognition systems, other scenarios can also benefit, such as telecommunications, military applications, medical applications, and law enforcement. In telecommunications, in addition to its potential to improve the telephone - based speech recognition performance, stress classification can be used to route 911 emergency call services for high priority emergency calls. The integration of speech recognition technology has already been seen in many military voice communication and control applications. Since many such applications involve stressful environments (e.g., aircraft cockpits, military peacekeeping/battlefield setting), stress classification and assessment become crucial to improve the system robustness in these applications. Finally, stress classification can also be employed in forensic speech analysis by law enforcement to assess the state of telephone callers or as an aid in suspect interviews.
Various techniques are used in the literature to classify emotional/stressed states on the basis of speech (for example Hidden Markov Models [3][6][7][5], Neural Networks [2][8][9]). In turn, these use different speech features, the most common being MFCC (Mel-Frequency Cepstral Coefficients), Pitch, LPC (Linear Prediction Coefficients), autocorrelation coefficients and Teager Energy Operator (TEO) based features [4].

Some researchers have combined various techniques to enhance performance in recognition of emotional states through speech, often using different speech features at the same time [3]. In [10] we proposed a genetic algorithm components selection approach to distinguish between positive and negative emotional states, but the aim of this paper is a broader classification, taking various speech styles into account.

2 Feature Extraction

The process whereby human speech is produced is typically modeled via a linear source-filter model. This model assumes that the flow of air propagates through the vocal tract as a plane wave. The vocal tract remains in a fixed configuration for short periods of time, so in each of these periods it can be considered as a time-invariant linear filter. Thus, in a long period of time the vocal tract is modelled as a time-variant linear filter. The speech features typically used in speech/speaker recognition processes and speech coding are obtained starting from identification of this filter. Some of these features have been studied for speech signal coding (*LPC, PARCOR, LSF, LAR*), while others are used for speech or speaker recognition (*MFCC, LPCC, Formants, Pitch, AC*).
Research focusing on the recognition of emotional states through speech has not up to now identified a specific feature.
Studies conducted by Teager [11] suggest the presence of vortices in the proximity of the vocal cords which interact with the primary flow and are the main source of excitement during closure of the cords. In addition, physiological changes in the vocal system caused by conditions of stress, for example muscular tension, will affect these interactions in the vocal tract [4]. These interactions are nonlinear, as confirmed by the theory of fluid mechanics [12] and numerical simulation of the Navier-Stokes equation [13]. It will thus be suitable to consider components extracted from nonlinear features in order to classify stress. In [4] the authors characterize the production of speech by modeling the configuration of the flow of air in the vocal tract and they propose three new features to explore the prospect of variations in the energy of airflow characteristics within the vocal tract. These features are the TEO-decomposed FM Variation (TEO-FM-Var), the normalized TEO Autocorrelation Envelope area (TEO-Auto-Env), and the Critical Band based TEO Autocorrelation Envelope area (TEO-CB-Auto-Env).
The idea on which this paper is based is to determine, starting from all the components obtained from a broad set of speech features, a subset of components that will make it possible to distinguish better between different emotional states than is possible when only a speech feature is used. In selecting these components we therefore introduced both the commonly used

features derived from the linear model and TEO-CB-Auto-Env feature which, as demonstrated in [4], is the nonlinear feature which allows for a better classification of different types of stress.
The speech was processed using a pre-emphasis filter to highlight the high-frequency components and then split into 30ms frames at a rate of 10ms. On the basis of the previous considerations, the following features were extracted from each frame:

- 4 LPC Spectrum based Formants (F_{1-4})
- 16 Mel-Frequency Cepstral based Coefficients ($MFCC_{1-16}$)
- 16 Real Cepstrum based coefficents ($RCEPS_{1-16}$)
- the Energy Level ($\log E$)
- autocorrelation based estimation of the Pitch (F_0)
- 17 Autocorrelation Coefficients (AC_{1-17})
- 16 Linear Prediction Coefficients (LPC_{1-16})
- 16 Reflection Coefficients ($PARCOR_{1-16}$)
- 16 Log Area Ratio Coefficients (LAR_{1-16})
- 16 Line Spectral Frequencies Coefficients (LSF_{1-16})
- 17 LPC Cepstral coefficients ($LPCC_{1-17}$)
- the Zero Crossing Rate (ZCR)
- the variance of the Linear Prediction Error ($\sigma^2_{E_{LPC}}$)
- 16 Critical Band Based Teager Energy Operator Autocorrelation Envelope Area ($TEO-CB-Auto-Env_{1-16}$)

The first- and second-order time differences were also computed as

$$\begin{aligned} \Delta x(n) &= x(n+1) - x(n-1) \\ \Delta^2 x(n) &= \Delta x(n+1) - \Delta x(n-1) \end{aligned} \tag{1}$$

obtaining for every frame a vector of 462 components.

3 Database

The extraction of speech features in the presence of different emotional states was performed using the SUSAS (Speech Under Simulated and Actual Stress) database [14], like in [4]. The database is partitioned into five domains, encompassing a wide variety of stresses and emotions. The five stress domains include:

i) talking styles (slow, fast, soft, loud, angry, clear, question);
ii) single tracking task or speech produced in noise (Lombard effect);
iii) dual tracking computer response task;

iv) actual subject motion-fear tasks (G-force, Lombard effect, noise, fear);
v) psychiatric analysis data (speech in states of depression, fear, anxiety).

The database contains both simulated speech under stress (*Simulated Domain*) and actual speech under stress (*Actual Domain*). The *Simulated Domain* consists of data from ten stressed styles (talking styles, single tracking task and Lombard effect domains); while the *Actual Domain* consists of speech produced while performing either (i) dual-tracking workload computer tasks, or (ii) subject motion-fear tasks (subjects in roller-coaster rides). The *Simulated Domain* uses recordings of 9 speakers in a quiet environment simulating speech under stress. The *Actual Domain* uses recordings of 7 speakers in states of actual roller coaster stress. In this work four different styles of speech from the *Simulated Domain* (*angry*, *loud*, *Lombard* and *neutral*) and two from the *Actual Domain* (*neutral* and *Roller Coaster stress*) were considered. A common highly confusable vocabulary set of 35 aircraft communication words makes up the SUSAS database. In the *Simulated Domain* each word in the vocabulary is repeated twice by each speaker. Not all the words in the vocabulary set are present in the *Actual Domain* and when they are they may not be repeated. For each of these speech styles a subset of words was chosen and then used in the feature selection, HMM training and test phases. To compare the stress recognition system with that presented in [4] we used the same words, i.e., "freeze", "help", "mark", "nav", "oh", "zero". Since the TEO is more applicable for voiced sounds than for unvoiced sounds, only high-energy voiced sections (i.e., vowels, diphthongs, liquids, glides, nasals) were extracted from the speech signal [4].

4 Selection of the Subset of Components

4.1 Separability criteria

Having chosen the classification system (neural network, HMM-based stochastic model, ...), one method that could be used to assess the validity of a subset of components would be to evaluate performance using the classification system itself. The time required to evaluate the various possible subsets of components would, however, be unacceptable due to the complexity of the algorithms that have to be used to train the classification system. It is necessary to define a criterion whereby it is possible to establish rapidly the degree of separability between L classes using a certain subset of components. In the *discriminant analysis* of statistics, within-class and between-class are used to formulate criteria of class separability [15]. A *within-class scatter-matrix* shows

the scatter of samples around their respective expected class vectors:

$$\mathbf{S_w} = \sum_{i=1}^{L} P_i E\left\{(\mathbf{X} - \mathbf{M_i})(\mathbf{X} - \mathbf{M_i})^T | \omega_i\right\} = \sum_{i=1}^{L} P_i \mathbf{R_i} \tag{2}$$

where: P_i is the a priori probability for class i, $\mathbf{X}$ is the parameter vector, $\mathbf{M_i}$ is the mean vector for class i, $\mathbf{R_i}$ is the covariance matrix for class i, ω_i represents class i, and L is the number of classes. The *between-classes scatter matrix* represents the scatter of the expected vectors around the mixture mean as

$$\mathbf{S_b} = \sum_{i=1}^{L} P_i (\mathbf{M_i} - \mathbf{M_0})(\mathbf{M_i} - \mathbf{M_0})^T \tag{3}$$

where $\mathbf{M_0} = E\{\mathbf{X}\} = \sum_{i=1}^{L} P_i \mathbf{M_i}$ represents the expected vector of the mixture distribution (i.e., the distribution of all the classes).
In order to formulate criteria for class separability, we need to convert the matrices to a number. This number should be larger when the between-class scatter is larger or the within-class scatter is smaller. There are several ways to do this. Given its simplicity of implementaiton, we used the following criterion

$$J_1 = tr(\mathbf{S_w}^{-1}\mathbf{S_b}) \tag{4}$$

where the symbol $tr(X)$ indicates the trace of the matrix X.

4.2 Selection Algorithm

The best subset of m components out of n may be found by evaluating a criterion of class separability for all possible combinations of m variables. However, the number of all possible combinations, $\binom{n}{m}$, becomes prohibitive even for modest values of m and n. For example, with $n = 24$ and $m = 12$ there are 2,704,156 possible combinations. It is therefore necessary to use techniques to avoid an exhaustive search. The techniques most widely used in the literature are *stepwise search techniques* such as the *backward selection (BS)* and *forward selection (FS)* procedures, *branch and bound methods (B&B)* and *stochastic global search methods* like *genetic algorithms (GAs)*.

4.2.1 Backward Selection

The *BS* procedure starts from the full set of n components. Then, eliminating one component, all possible subsets of $n - 1$ components are obtained and their criterion values are evaluated. The highest value is determined and the corresponding subset is selected as the best of those with $n - 1$ components. Another component is then eliminated from this subset and the best subset

with $n-2$ components is determined. The procedure is repeated until the best subset containing the number m of desired components is obtained.

4.2.2 Forward Selection

The *FS* procedure starts by evaluating the separability criterion for each component. The highest value is determined and the corresponding component is selected as the best. All possible pairs of components which contain this component are established and their separability criterion determined. The pair with the highest value is selected as the best containing 2 components. The procedure is repeated until the best subset containing the number m of desired components is obtained.
Both *BS* and *FS* evaluate the increase in performance obtained by eliminating or adding each component and so, although they are simple search techniques, they do not always achieve the best solution.

4.2.3 Branch and Bound

A *B&B* algorithm searches the complete space of solutions for the best solution to a given problem. Some solutions are not actually explored because they are known a priori not to be optimal. When it becomes apparent that a solution is not optimal exploration of it is abandoned (bound). The use of bounds for the function to be optimized combined with the value of the current best solution enables the algorithm to search parts of the solution space only implicitly. The order in which solutions are explored is important: the sooner a good solution is found, the more effective the bound conditions will be later, thus reducing exploration costs. Despite investing in the search for exact algorithms that are capable of solving the problem of parameter selection, their complexity will always grow exponentially along with the number of components to be selected.
Given the large number of components used in our approach, it was not possible to use *B&B* algorithms.

4.2.4 General Remarks on Genetic Algorithms

A *GA* is a stochastic global search method that mimics the metaphor of natural biological evolution. Problems which appear to be particularly appropriate for solution by genetic algorithms include timetabling and scheduling problems. GAs have also been applied to engineering and to solving global optimization problems. GAs operate on a population of potential solutions applying the principle of the survival of the fittest to produce (hopefully) better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to

their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals they were created from, just as in natural adaptation. Individuals, or current approximations, are encoded as strings, *chromosomes*, composed over an alphabet, so that the *genotypes* (chromosome values) are uniquely mapped onto the decision variable (*phenotypic*) domain. Having decoded the chromosome representation into the decision variable domain, it is possible to assess the performance, or fitness, of individual members of a population. This is done through an objective function that characterizes an individual's performance in the problem domain. Thus, the objective function establishes the basis for selection of pairs of individuals that will be mated together during reproduction. During the reproduction phase, each individual is assigned a fitness value derived from its raw performance measure given by the objective function. This value is used in the selection to bias towards fitter individuals. Highly fit individuals, relative to the whole population, have a high probability of being selected for mating whereas less fit individuals have a correspondingly low probability of being selected. Once the individuals have been assigned a fitness value, they can be chosen from the population, with a probability according to their relative fitness, and recombined to produce the next generation. Selection is the process of determining the number of times, or trials, a particular individual is chosen for reproduction and, thus, the number of offspring that an individual will produce. A real-valued interval, Sum, is determined as the sum of the raw fitness values over all the individuals in the current population. Individuals are then mapped one-to-one into contiguous intervals in the range $[0, Sum]$. To select an individual, a random number is generated in the interval $[0, Sum]$ and the individual whose segment spans the random number is selected. This process is repeated until the desired number of individuals have been selected.

Genetic operators manipulate the characters (genes) of the chromosomes directly, using the assumption that certain individual's gene codes, on average, produce fitter individuals. The recombination (or crossover) operator is used to exchange genetic information between pairs, or larger groups, of individuals. This crossover operation is not necessarily performed on all strings in the population. Instead, it is applied with a probability $Prob_{cross}$ when the pairs are chosen for breeding. A further genetic operator, called mutation, is then applied to the new chromosomes, again with a set probability, $Prob_{mut}$. Mutation causes the individual genetic representation to be changed according to some probabilistic rule. Mutation is generally considered to be a background operator that ensures that the probability of searching a particular subspace of the problem space is never zero. This has the effect of tending to inhibit the possibility of converging to a local optimum, rather than the global optimum. After recombination and mutation, the individual strings are then, if necessary, decoded, the objective function evaluated, a fitness value assigned to each individual and individuals selected for mating according to their fitness,

and so the process continues through subsequent generations. In this way, the average performance of individuals in a population is expected to increase, as good individuals are preserved and bred with one another and the less fit individuals die out. The GA is terminated when some criteria are satisfied, e.g., a certain number of generations, a mean deviation in the population, or when a particular point in the search space is encountered.

4.2.5 Genetic Algorithms for Components Selection

Genetic algorithms have demonstrated substantial improvement over a variety of random and local search methods. This is accomplished by their ability to exploit accumulating information about an initially unknown search space in order to bias subsequent search into promising subspaces. Since GAs are basically a domain-independent search technique, they are ideal for applications where domain knowledge and theory is difficult or impossible to provide [16]. For GAs to work the number of components to be selected must be predetermined and constant. It is therefore necessary to modify the typical functioning of GAs so that this constraint is met. Let n be the total number of components available to choose from to represent the patterns to be classified. Each subset of components is a *chromosome* and is represented by a binary vector of size $L_{ind} = n$. If a bit is a 1, it means that the corresponding component is selected. A value of 0 indicates that the corresponding component is not selected. GAs operate simultaneously on a number of potential solutions, called a population, consisting of some encoding of the parameter set. The initial population is achieved by generating the required number of individuals using a random number generator that uniformly distributes numbers in the desired range. With a binary population of N_{ind} individuals whose chromosomes are L_{ind} bits long, $N_{ind} \cdot L_{ind}$ random uniformly distributed numbers from the set $0, 1$ would be produced, such that the number of $1s$ in each row is equal to m. The algorithm generating the initial population is shown in Fig. 1. The function *randperm(n)* returns a random permutation of the first n integers.

The objective function used to run the GA was equal to the inverse of the separation index J_1^{-1}. In our approach the number of individuals in the new population will be equal to the number of individuals in the initial population N_{ind}. The crossover operation is applied with a probability of $Prob_{cross} = 0.7$ when the pairs are chosen for breeding. This crossover probability value is typically used in GAs and generally yields good results in terms of convergence towards the optimal solution. Our system uses the simplest form of crossover called *single-point crossover*. Fig. 2 shows the algorithm used for recombination. Let O and E be the arrays containing the indexes of the components selected for the parents; having generated a random floating number between 0 and 1 (*rand(1.0)* function) recombination is only performed when this number is lower than the pre-established $Prob_{cross}$. An integer position x is selected uniformly at random between 1 and the string length m (indicated in Fig. 2

by means of the *randint* function), and the genetic information exchanged between the individuals about this point; then two new offspring strings O^* and E^* are produced. When the parents have components in common, the offspring may have fewer than m components selected. For this reason a *check routine* illustrated in Fig. 3 is used, which ensures offspring with the pre-established number of features, m. This is achieved by exploiting the components not shared by the parents and the offspring produced (in the algorithm in Fig. 3 the "\" operator yields all indexes in the array that appear in the first operand but not in the second).

The mutation algorithm is applied in such a way that it can be verified with a probability of $Prob_{mut} = 0.7$ for each member of the population. As with crossover, the value selected for the mutation probability is one typically used in GAs as it gives good results in terms of convergence towards the optimal solution. When one or more members invert their value, passing from 0 to 1 or 1 to 0, the number of elements with a value of 1 must be equal to m. Once again the check routine in Fig. 3 is used.

For each generation cycle the positions of the $1s$ in the row with the lowest objective function value indicate the m best components for each generation. The generational cycle is repeated 300 times and at each generation the system stores the set of m components with the best performance in terms of the separation index. At the end of the generational cycle the set chosen is the one with the best separation index.

Fig. 4 is an example of the trend followed by the separation index (the inverse of the objective function) as the number of generation cycles progresses.

4.3 Genetic Algorithm features

At this point it is necessary to establish the number of components to be extracted during the selection procedure. This is important so as to be able to make significant comparisons with the results obtained using other feature vectors. The results in [4] were obtained using HMM models trained with 16 MFCCs feature vector and TEO-CB-Auto-Env feature vector of 16 components to classify between the 4 states of the *Simulated Domain* and between the 2 states of the *Actual Domain*. We therefore initially extracted exclusively 16 components *(16-GA feature vector)* from the set of non time-derivative features. Then, to obtain even better performance, we introduced the time-derivative features and determined the separation index using vectors with a larger number of components. As the number of components used increases, the speed at which the separation index grows decreases. It was found that when more than 48 components are used the increase in the separation index is insignificant. We therefore extracted 48 components *(48-GA feature vector)* from the set of all features. As an example, Table 1 indicates the 48 components selected using the GA technique. The first column in the table

Table 1
48-GA Feature Vector

Feature	**Components Selected** Δ^0	Δ^1	Δ^2	#
AC_{1-17}	1	15	-	2
F_0	1	-	1	2
F_{1-4}	1,2,3,4	2,4	2,4	8
LAR_{1-16}	3	-	4	2
$\log E$	1	1	-	2
LPC_{1-16}	-	-	2	1
$LPCC_{1-17}$	3,4,5,7,15	-	13	6
LSF_{1-16}	1,5,12	-	1	4
$MFCC_{1-16}$	9	-	-	1
$PARCOR_{1-16}$	1,2,3,4,6,9,11	-	1,11	9
$RCEPS_{1-16}$	1,2,8	-	-	3
TEO_{1-16}	1,3,7,11	-	8,12,16	7
$\sigma^2_{E_{LPC}}$	1	-	-	1
ZCR	-	-	-	0

(*Feature*) contains the various features used; the second column (Δ^0) contains the indexes of the components selected from the non time-derivative features; the third (Δ^1) and fourth (Δ^2)) columns respectively contain the indexes of the components selected from the first and second derivative of the features; the last column (#) gives the total number of components selected for each feature. It should be pointed out that a single component selected by a GA is not significant as the objective function is evaluated over the whole subset. More specifically, if the selection using GA is repeated, their statistical nature may yield a different combination of components, but (in the optimal case) it would still converge on the same objective function value. In this case the aim of selection is to identify a subset with a high degree of separability and which is assumed to give better performance when used as input for a classification system.

5 Evaluations

In [10] we used two suboptimal techniques for components selection, the FS and the GA selection approach, and we showed that the performance obtained with the selection technique based on GA was consistently better than that of the FS technique. Thus the performance that can be obtained through the GA feature vectors was compared with that of the linear features considered to be most efficient at recognizing a speaker's emotional state, MFCCs and Pitch [2][17], and the nonlinear feature TEO-CB-Auto-Env [4]. To compare the performance of the system proposed with that of others in the literature, as was done in [4], we performed the evaluation in three different contexts:

- Text-Dependent Pairwise Stress Classification
- Text-Independent Pairwise Stress Classification
- Text-Independent Multistyle Stress Classification

The classifier used in the test was a baseline five-state HMM-based classifier with continuous distributions, each with two Gaussian mixtures. The HMMs were trained and tested using the Hidden Markov Model ToolKit (HTK-3.3) [18]. To compare the performance obtainable using the different types of features as input to the classification systems, the HMMs were trained and tested with the following inputs:

a) *MFCCs feature vector* (16 components);
b) *Pitch feature* (scalar);
c) *TEO-CB-Auto-Env feature vector* (16 components);
d) *16-GA feature vector* (16 components);
e) *48-GA feature vector* (48 components).

5.1 Text-Dependent Pairwise Stress Classification

The first step involved text-dependent pairwise classification, in which the HMMs were trained and tested with the same words. An HMM was trained with the voiced part of each of the words from each style of speech chosen for the training phase. There are thus 24 HMMs (6 words x 4 styles of speech) for the *Simulated Domain* and 12 (6 words x 2 styles of speech) for the *Actual Domain*. The HMMs were trained with a series of "replicas" of the same word uttered by various speakers. Due to the low number of voice tones available for pairwise classification, the "round-robin" method used in [4] was applied (e.g., in the *Simulated Domain* for each of the 18 "replicas" of a word the relative HMM is trained with 17 of the replicas and tested with the remaining word). The results of this classification are shown in Fig. 5. From analysis of the figure it emerges that when the *GA features* (both 16 and 48 components)

are used the results are on average better than in all the other cases. Using the *16-GA feature* there is an average increase in performance of about 5% as compared with the results obtained using MFCCs, about 7% as compared with the results obtained using TEO-CB-Auto-Env and about 4% as compared with the results obtained using Pitch. Analyzing the standard deviation of the classification obtained with the different speech styles it is observed that all the types of features maintain the same consistency. Only in the case of classification between *loud* and *neutral* better performance was achieved using MFCCs feature vector.

5.2 Text-Independent Pairwise Stress Classification

The second test involved text-independent pairwise classification to see whether the performance of these features depends, and to what extent, on the information contained in a text or phoneme. A single HMM was trained for each style of speech in the two domains: for the *Simulated Domain* four HMMs were trained with 108 words belonging to the four styles, whereas 270 different words were used in the test phase. For the *Actual Domain* the two HMMs for the *neutral* and *stressed* styles were trained with 94 words each and the tests were performed using 140 different words. Fig. 6 shows the results of this classification. The results obtained using *16-GA feature* were on average slightly better than those obtained using TEO-CB-Auto-Env (the average increase in performance is about 3%). The performance obtained using both MFCCs and Pitch decreases significantly in this context due to their dependence on the phonetic content of the words. Text-independent classification using *48-GA feature* performed very well with regard to the pairs belonging to the *Simulated Domain*, and also in the *Actual Domain* performance was clearly better than that achieved using the other features. When the *48-GA feature* was used the average increase in performance as compared with the TEO-CB-Auto-Env is about 10%. It was also observed that in this case the consistency of the features was more differentiated: the standard deviation of the classificatrion results obtained using *48-GA feature* was less than 1%; It was about 4% when TEO-CB-Auto-Env was used, and over 10% when Pitch and MFCCs were used.

5.3 Text-Independent Multistyle Stress Classification

The aim of the last phase was multistyle text-independent stress classification. The aim was to verify the accuracy of the features in distinguishing between neutral and stress-affected speech, and then to evaluate their efficiency in classifying various types of stress. The *Actual Domain* was not considered in

Table 2
Text-Independent multistyle classification using MFCCs.

Test Speech Style	**Distribution of Speech Style Detection Rate (%)**				**Neutral-Stressed Detection Rate (%)**	
	Neutral	**Angry**	**Loud**	**Lombard**	**Neutral**	**Stressed**
Neutral	**78.3**	7.66	9.6	4.44	**78.3**	21.7
Angry	20.3	**46.86**	21.36	11.48	20.3	**79.7**
Loud	20.84	31	**33.32**	14.84	20.84	**79.16**
Lombard	29.9	20.63	29.74	**19.73**	29.9	**70.1**

Table 3
Text-Independent multistyle classification using PITCH.

Test Speech Style	**Distribution of Speech Style Detection Rate (%)**				**Neutral-Stressed Detection Rate (%)**	
	Neutral	**Angry**	**Loud**	**Lombard**	**Neutral**	**Stressed**
Neutral	**53.24**	1.8	3.6	41.36	**53.24**	46.76
Angry	14.73	**41.15**	18.37	25.75	14.73	**85.27**
Loud	13.1	33.65	**34.35**	18.9	13.1	**86.9**
Lombard	8.42	6.9	7.27	**77.41**	8.42	**91.58**

this phase as the stress present in the voice tones in this domain is strong and less easy to detect in most real cases. Each of the 270 words outside the vocabulary used in the Text-Independent Pairwise test phase was classified using the four HMMs for the four speech styles in the *Simulated Domain*. The output was therefore not simply words classified as neutral or stressed but as belonging to one of the four styles of stress considered.
The results obtained with the various features are given in Tables 2, 3, 4, 5 and 6. The first part of each table is the matrix of misclassification between the various speech styles. So if, for example, the system input is presented with a token belonging to the neutral class, classification is correct only if the neutral model achieves maximum verisimilitude. The second part of the tables gives the Neutral-Stressed Detection Rate: if the input is a token belonging to one or other of the *angry*, *loud* and *Lombard* classes, the token will be correctly classified if any one of the three models (*angry*, *loud* or *Lombard*) obtains the maximum verisimilitude. In this way it is possible to compare the results in the various tables directly. Analysis of the second part of the tables shows that when GA features are used performance is considerably better in classification of the *neutral* style. When models trained with *48-GA feature* are used the *neutral* style is always recognized correctly.
To analyze the results obtained in classification between the 4 different speech

Table 4
Text-Independent multistyle classification using TEO-CB-AUTO-ENV.

Test Speech Style	Distribution of Speech Style Detection Rate (%)				Neutral-Stressed Detection Rate (%)	
	Neutral	Angry	Loud	Lombard	Neutral	Stressed
Neutral	**73.55**	4.32	2.1	20.03	**73.55**	26.45
Angry	7.4	**62.24**	14.81	15.55	7.4	**92.6**
Loud	0.74	36.03	**35.23**	28	0.74	**99.26**
Lombard	15.55	8.91	8.15	**67.39**	15.55	**84.45**

Table 5
Text-Independent multistyle classification using 16-GA feature.

Test Speech Style	Distribution of Speech Style Detection Rate (%)				Neutral-Stressed Detection Rate (%)	
	Neutral	Angry	Loud	Lombard	Neutral	Stressed
Neutral	**81.71**	6.62	3.31	8.36	**81.71**	18.29
Angry	6.8	**65.6**	14.55	13.05	6.8	**93.20**
Loud	3.7	29.6	**51.6**	15.1	3.7	**96.30**
Lombard	8.89	16.83	9.08	**65.2**	8.89	**91.11**

Table 6
Text-Independent multistyle classification using 48-GA feature.

Test Speech Style	Distribution of Speech Style Detection Rate (%)				Neutral-Stressed Detection Rate (%)	
	Neutral	Angry	Loud	Lombard	Neutral	Stressed
Neutral	**100**	0	0	0	**100**	0
Angry	9.04	**75.4**	3.04	12.52	9.04	**90.96**
Loud	10.3	0	**64.62**	25.08	10.3	**89.7**
Lombard	0	3.78	5.27	**90.95**	0	**100**

styles we can use the values on the diagonal of the misclassification matrix in the first part of each table. As the results show, there is once again an increase in performance when the GA features are used. Performance analysis shows that when TEO-CB-Auto-Env feature vector is used classification between the 4 speech styles is on average better than when MFCCs feature vector and Pitch are used. The MFCCs feature performs better, however, in classifying the *neutral* state and Pitch performs better when classifying the *Lombard* state. Comparing the results obtained using *16-GA feature* with those obtained using

TEO-CB-Auto-Env, there is an improvement of about 8% in the *neutral* case, about 3% in the *angry* case, 16% in the *loud* case and a slight deterioration of about 2% in the *Lombard* case. As can be seen from Table 6 the best performance in classification of the 4 speech styles is obtained using the *48-GA feature*.

6 Conclusions and Future Work

The paper has proposed a GA-based components selection procedure to build new speech features for distinguishing between different styles of stress. It has been demonstrated that the recognition system using these GA features performed better than the others in three different evaluations: Text-Dependent Pairwise Stress Classification, Text-Independent Pairwise Stress Classification and Text-Independent Multistyle Stress Classification.
Rather than recognizing emotional states from the way a single word is uttered, the authors think that better results could be obtained by analyzing whole sentences uttered under a given type of stress. Sentences could be divided into time segments of finite duration and the technique could then be applied to each segment. These time segments have to be chosen very carefully as they have to fulfill two conflicting conditions:
1) emotional changes can occur very quickly, but the segment length sets the temporal resolution of recognizable changes,
2) reliable statistical features can often only be computed over longer segments.

Acknowledgment

The authors would like to thank TIM (Telecom Italia Mobile) for supporting this work. The anonymous reviewers and the Co Editor-in-Chief Renato De Mori are gratefully acknowledged for very helpful comments, which considerably improved the presentation.

References

[1] R. Cowie, E. Douglas-Cowie, N. Tsapatsoulis, G. Votsis, S. Kollias, W. Fellenz, J. Taylor, Emotion Recognition in Human-Computer Interaction, IEEE Signal Processing Magazine 18 (1) (2001) 32–80.

[2] J. H. L. Hansen, B. Womack, Feature Analysis and Neural Network-Based Classification of Speech Under Stress, IEEE Transactions on Speech and Audio Processing 4 (4) (1996) 307–313.

[3] S. E. Bou-Ghazale, J. H. L. Hansen, A Comparative Study of Traditional and Newly Proposed Features for Recognition of Speech Under Stress, IEEE Transactions on Speech and Audio Processing 8 (4) (2000) 429–442.

[4] G. Zhou, J. H. L. Hansen, J. F. Kaiser, Nonlinear Feature Based Classification of Speech Under Stress, IEEE Transactions on Speech and Audio Processing 9 (3) (2001) 201–216.

[5] T. L. Nwe, S. W. Foo, L. De Silva, Classification of Stress in Speech Using Linear and Nonlinear Features, in: Proceedings of 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 2, pp. 9–12, April 6-10, 2003, Hong Kong.

[6] B. Schuller, G. Rigoll, M. Lang, Hidden Markov Model-Based Speech Emotion Recognition, in: Proceedings of 2003 International Conference on Multimedia and Expo (ICME '03), Vol. 1, pp. 401–404, July 6-9, 2003, Baltimore, Maryland USA.

[7] T. L. Nwe, F. S. Wei, L. De Silva, Speech Based Emotion Classification, in: Proceedings of IEEE Region 10 International Conference on Electrical and Electronic Technology (TENCON), Vol. 1, pp. 297–301, August 19-22, 2001, Phuket Island, Langkawi Island, Singapore.

[8] C.-H. Park, K.-B. Sim, Emotion Recognition and Acoustic Analysis from Speech Signal, in: Proceedings of the International Joint Conference on Neural Networks, Vol. 4, pp. 2594–2598, July 20-24, 2003, Portland, Oregon, USA.

[9] J. Nicholson, K. Takahashi, R. Nakatsu, Emotion Recognition in Speech Using Neural Networks, in: Proceedings of 6th International Conference on Neural Information Processing, Vol. 2, pp. 495–501, November 16-20, 1999, Perth, W.A.

[10] F. Beritelli, S. Casale, A. Russo, S. Serrano, A Genetic Algorithm Feature Selection Approach to Robust Classification between "Positive" and "Negative" Emotional State in Speakers, in: Procedings of 39th Annual Asilomar Conference on Signals, Systems, and Computers, pp. 550–553, October 29 - November 1, 2005, Pacific Grove, California, USA.

[11] H. Teager, Some Observations on Oral Air Flow During Phonation, IEEE Transactions on Acoustics, Speech, and Signal Processing 28 (5) (1980) 599–601.

[12] A.J.Chorin, J.E.Marsden, A Mathematical Introduction to Fluid Mechanism, 2nd Edition, Springer-Verlag, 1990.

[13] T.J.Thomas, A Finite Element Model of Fluid Flow in the Vocal Tract, Computer Speech Language 1 (1986) 131–151.

[14] J. Hansen, S. Bou-Ghazale, Getting Started with SUSAS: A Speech Under Simulated and Actual Stress Database, in: Proceedings of International Conference on Speech Communication and Technology (Eurospeech), Vol. 4, pp. 1743–1746, September 22-25, 1997, Rhodes, Greece.

[15] K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, 1990, Ch. 10, pp. 446–448.

[16] H. Vafaie, K. D. Jong, Robust Feature Selection Algorithms, in: Proceedings of the Fifth International Conference on Tools with Artificial Intelligence. TAI '93, pp. 356–363, November 8-11, 1993 - Boston, MA, USA.

[17] B.D.Womack, J.H.L.Hansen, Classification of Speech Under Stress Using Target Driven Features, Speech Communications 20 (1996) 131–150.

[18] S. Young, et al, The HTK Book (for HTK Version 3.3), Cambridge University Engineering Department (2005).

```
ROUTINE CREATE INITIAL POPULATION
INPUT:
nothing
OUTPUT:
initial population of chromosomes C

C_{i,j}=0 : i=1..NIND, j=1..n
I = randperm(n)
j_1 = 1
j_2 = m
i = 1
while i <= NIND
   C_{i,I[j1..j2]}=1
   j_1 = j_1 + m
   j_2 = j_2 + m
   if j_2 > n
      if j_1 < n
         i = i + 1
         C_{i,I[j1..n]}=1
         C_{i,I[1:j2-n]}=1
      end
      I = randperm(n)
      j_1 = 1
      j_2 = m
   end
   i = i + 1
end
```

Fig. 1. Algorithm used to create the initial population.

```
ROUTINE CROSSOVER
INPUT:
selected individuals C
OUTPUT:
new individuals C*

C*_{i,j}=0 : i=1..N_ind, j=1..n
while i <= N_ind
   if rand(1.0) < Prob_cross
      k=1, h=1
      for j=1...n
         if C_{i,j}==1
            O_h = j
            h=h+1
         end
         if C_{i+1,j}==1
            E_k = j
            k=k+1
         end
      end
      x = randint(m)
      O*=[O_1 ...O_x  E_{x+1} ...E_m]
      E*=[E_1 ...E_x  O_{x+1} ...O_m]
      O* = check(O*, i, O, E)
      E* = check(E*, i+1, O, E)
      for h=1...m
         C*_{i,O*_h}=1
         C*_{i+1,E*_h}=1
      end
   end
   i = i + 2
end
```


Fig. 2. Algorithm used for crossover.

```
ROUTINE CHECK
INPUT:
offspring index array X
chromosome position p
first parent index array P1
second parent index array P2
OUTPUT:
checked offspring index array X

Cj=0 : j=1..n
 for h=1...m
    C_X(h) = 1
 end
if sum_{j=1}^{n} Cj < m
   X*=sort(X)
   A = [P1 P2]
   D = A \ X*
   I = randperm(length(D))
   k=1
   for h = 1..m
      if X*_h == X*_{h+1}
         X*_h = D(I(k))
         k=k+1
         if k > length(D) k = 1
      end
   end
end
X = X*
```

Fig. 3. Algorithm used to maintain a constant number of components selected after crossover or mutation.

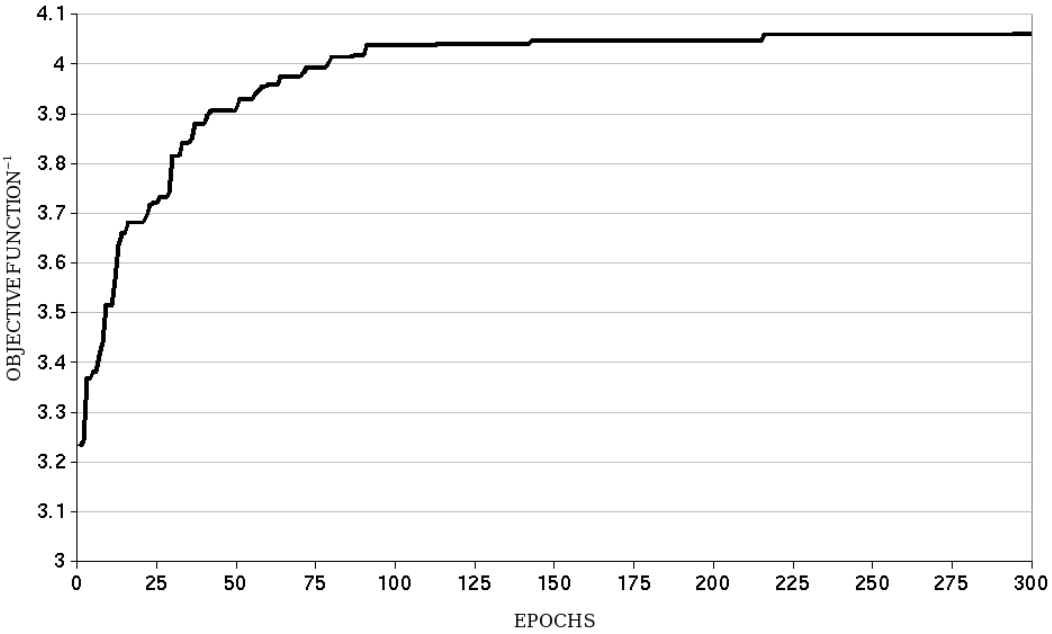


Fig. 4. Example of objective function trend.

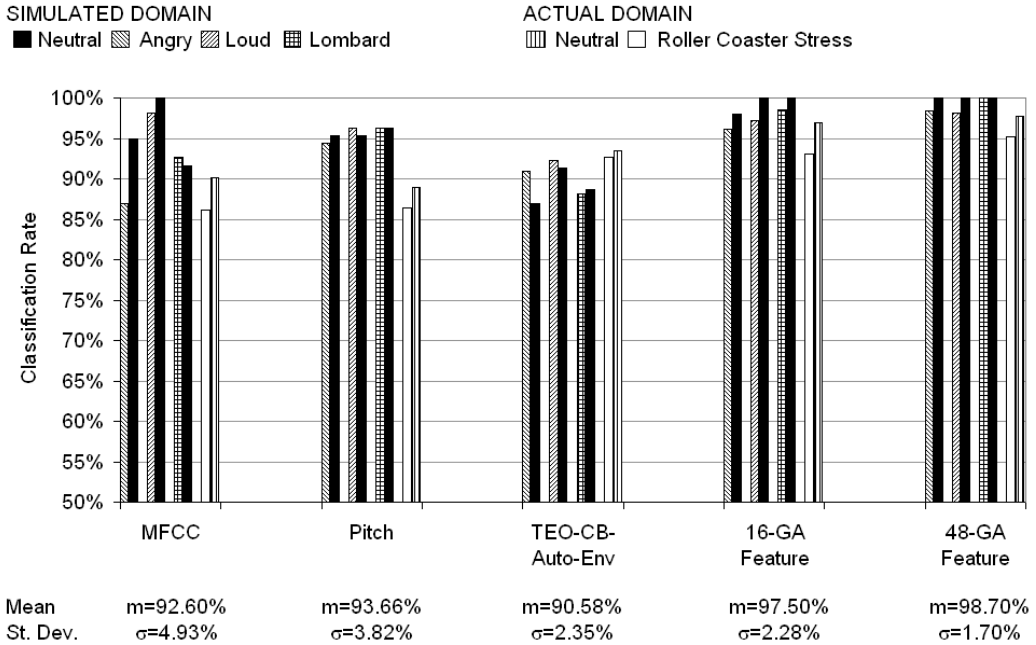


Fig. 5. Text-Dependent pairwise stress classification results.

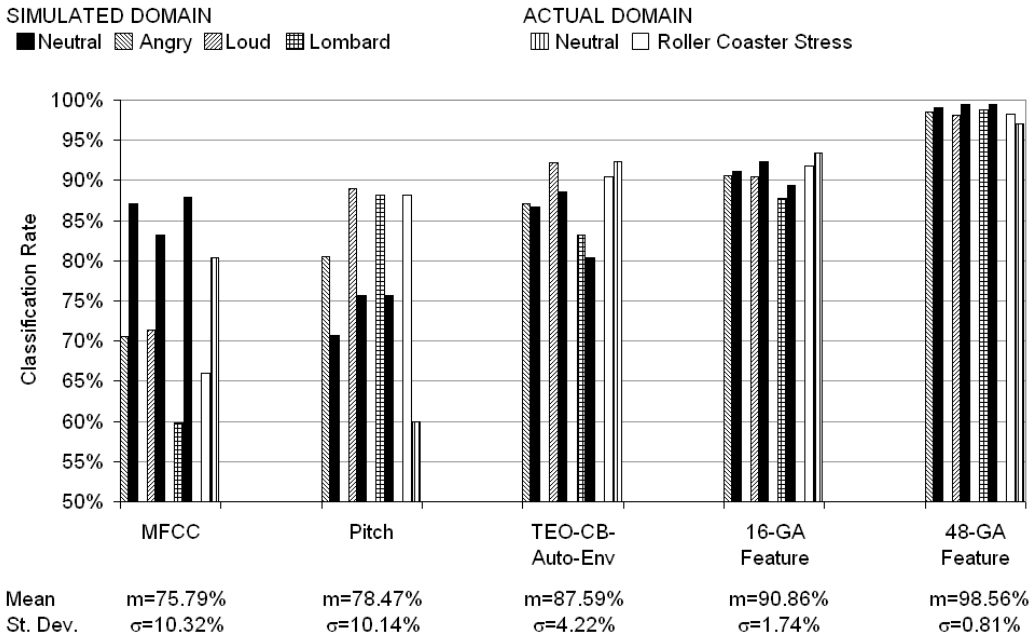


Fig. 6. Text-Independent pairwise stress classification results.