Detecting Autism, Emotions and Social Signals Using AdaBoost

Gábor Gosztolya[1], *Róbert Busa-Fekete*[1,2], *László Tóth*[1]

[1]Research Group on Artificial Intelligence, Szeged, Hungary
[2] Department of Mathematics and Computer Science, University of Marburg, Germany
{ ggabor, busarobi, tothl } @inf.u-szeged.hu

Abstract

In the area of speech technology, tasks that involve the extraction of non-lingustic information have been receiving more attention recently. The Computational Paralinguistics Challenge (ComParE 2013) sought to develop techniques to efficiently detect a number of paralinguistic events, including the detection of non-linguistic events (laughter and fillers) in speech recordings as well as categorizing whole (albeit short) recordings by speaker emotion, conflict or the presence of development disorders (autism). We treated these sub-challenges as general classification tasks and applied the general-purpose machine learning meta-algorithm, AdaBoost.MH, and its recently proposed variant, AdaBoost.MH.BA, to them. The results show that these new algorithms convincingly outperform baseline SVM scores.
Index Terms: speech recognition, speech technology, emotion detection, machine learning, AdaBoost.MH, AdaBoost.MH.BA

1. Introduction

Though speech recognition is undoubtedly a machine learning task, over the decades researchers developed dedicated tools for it, most notably the hidden Markov model (HMM) [1]. This is mainly because speech recognition is special in the sense that it has to handle a sequence of samples jointly, rather than simply classifying separate samples. Hence, general-purpose machine learning algorithms are rarely used for speech recognition, and when they are, they are mostly applied only for phoneme classification. In this task, in a standard frame-based framework each small portion of the speech signal has to be identified as one of the possible phonemes in the given language. A second reason is that, compared to usual learning tasks, speech recognition databases have somewhat specific database characteristics from a machine learning perspective. First, there are a moderate number of classes (e.g. phonemes distinguished in the given language). Second, there are usually few features: the very popular MFCC + Δ + $\Delta\Delta$ feature set (containing the mel-frequency spectral coefficients, their first and second order derivatives) consists of only 39 features for a frame. This set can be expanded with some other attributes; still, the size of the feature set rarely exceeds a few hundred. But most importantly, a typical speech recognition database contains at least several hours of recordings (nowadays hundreds of hours is typical), which leads to literally millions of training examples. This setup is quite uncommon in machine learning, though, and most classification algorithms are not prepared to work with such a huge amount of data. It is more typical to have significantly fewer training examples that are described by a large feature vector, and which belong to one of the few classes, like in [2]. Such tasks arise in speech technology as well, though; e.g. in speaker recognition [3] and emotion detection [4, 5].

The sub-challenges of the Computational Paralinguistics Challenge (ComParE 2013) [6], in general, are closer to this latter scenario, as the task is to classify *the whole recording*. In the **Conflict Sub-Challenge**, group discussions have to be evaluated by seeking to retrieve conflicts. In the **Emotion Sub-Challenge**, the emotional state of the speaker has to be identified, along with its arousal and valence (being either low or high). In the **Autism Sub-Challenge**, the type of pathology of the speaker has to be determined. In each of these sub-challenges, besides the wave files, a pre-extracted 6373-long feature vector per recording was also available, but the number of training samples was relatively small.

The **Social Signals Sub-Challenge** is quite different from the other three. In it, non-linguistic events (*fillers* and *laughter*) need to be distinguished from speech and normal pause (collectively called *garbage*) for each 10ms-long sample (*frame*) of a recording. Unlike the former sub-challenges, the feature set is also smaller, there being only 141 attributes for each frame.

We viewed these tasks as classification tasks, but the Conflict Sub-Challenge was more like a regression [7] task as we are expected to match a given *numeric score* instead of a class label, the latter being determined by the signum of the score. Despite getting fairly good results in finding the class label, in the end we did not participate in this sub-challenge.

With the partial exception of the Social Signals Sub-Challenge, we treated these tasks as pure classification tasks, and applied a general-purpose machine learning algorithm to classify the samples. We focused our investigations on the feature vector given by the organizers, and did not perform any kind of feature extraction from the waveforms themselves. The method applied was AdaBoost, which is an iterative learning meta-algorithm. It combines a large number of base learners, their number being equal to the number of training iterations. This algorithm was reported to perform well on various classification tasks [8, 9].

2. Learning with AdaBoost

For the formal description of the machine learning task, let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be the $n \times d$ *observation matrix*, where $x_i^{(j)}$ are the elements of the d-dimensional observation vectors $\mathbf{x}_i \in \mathbb{R}^d$. We are also given a *label matrix* $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ of dimension $n \times K$ where $\mathbf{y}_i \in \{+1, -1\}^K$. In a *multi-class* classification one and only one of the elements of $\mathbf{y}_i$ is $+1$, whereas in a *multi-label* (or *multi-task*) classification $\mathbf{y}_i$ is arbitrary, as $\mathbf{x}_i$ can belong to several classes at the same time. In the former case we denote the index of the correct class by $\ell(\mathbf{x}_i)$.

This publication is supported by the European Union and co-funded by the European Social Fund. Project title: Telemedicine-focused research activities in the fields of mathematics, informatics and medical sciences. Project number: TÁMOP-4.2.2.A-11/1/KONV-2012-0073

Copyright © 2013 ISCA

AdaBoost.MH($\mathbf{X}, \mathbf{Y}, \mathbf{W}^{(1)}$, Base$(\cdot,\cdot,\cdot), T$)

1 **for** $t \leftarrow 1$ **to** T

2 $\quad \mathbf{h}^{(t)}(\cdot) \leftarrow \alpha^{(t)}\mathbf{v}^{(t)}\varphi^{(t)}(\cdot) \leftarrow$ Base$\big(\mathbf{X}, \mathbf{Y}, \mathbf{W}^{(t)}\big)$

3 $\quad$ **for** $i \leftarrow 1$ **to** n **for** $\ell \leftarrow 1$ **to** K

4 $$\quad w_{i,\ell}^{(t+1)} \leftarrow w_{i,\ell}^{(t)} \frac{\exp\big(-h_\ell^{(t)}(\mathbf{x}_i) y_{i,\ell}\big)}{\sum_{i'=1}^{n}\sum_{\ell'=1}^{K} w_{i',\ell'}^{(t)} e^{-h_{\ell'}^{(t)}(\mathbf{x}_{i'}) y_{i',\ell'}}}$$

5 **return** $\mathbf{f}^{(T)}(\cdot) = \sum_{t=1}^{T} \mathbf{h}^{(t)}(\cdot)$

Figure 1: The pseudocode for the AdaBoost.MH algorithm. $\mathbf{X}$ is the observation matrix, $\mathbf{Y}$ is the label matrix, $\mathbf{W}^{(1)}$ is the initial weight matrix, Base$(\cdot,\cdot,\cdot)$ is the base learner algorithm, and T is the number of iterations. $\alpha^{(t)}$ is the base coefficient, $\mathbf{v}^{(t)}$ is the vote vector, $\varphi^{(t)}(\cdot)$ is the scalar base classifier, $\mathbf{h}^{(t)}(\cdot)$ is the vector-valued base classifier, and $\mathbf{f}^{(T)}(\cdot)$ is the final (strong) classifier.

2.1. AdaBoost.MH

The goal of the AdaBoost.MH algorithm [10] (see Figure 1) is to return a vector-valued classifier $\mathbf{f} : \mathcal{X} \to \mathbb{R}^K$ with a small *Hamming loss*

$$R_{\mathrm{H}}\big(\mathbf{f}^{(T)}, \mathbf{W}^{(1)}\big) = \sum_{i=1}^{n}\sum_{\ell=1}^{K} w_{i,\ell}^{(1)} \mathbb{I}\{\mathrm{sign}\big(f_\ell^{(T)}(\mathbf{x}_i)\big) \neq y_{i,\ell}\}^1$$

by minimizing its upper bound (the exponential margin loss)

$$R_{\mathrm{e}}\big(\mathbf{f}^{(T)}, \mathbf{W}^{(1)}\big) = \sum_{i=1}^{n}\sum_{\ell=1}^{K} w_{i,\ell}^{(1)} \exp\big(-f_\ell^{(T)}(\mathbf{x}_i) y_{i,\ell}\big), \quad (1)$$

where $f_\ell(\mathbf{x}_i)$ is the ℓth element of $\mathbf{f}(\mathbf{x}_i)$. The user-defined weights $\mathbf{W}^{(1)} = \big[w_{i,\ell}^{(1)}\big]$ are usually set either uniformly to $w_{i,\ell}^{(1)} = 1/(nK)$, or, in multi-class classification, to

$$w_{i,\ell}^{(1)} = \begin{cases} \frac{1}{2n} & \text{if } \ell = \ell(\mathbf{x}_i) \text{ (i.e., if } y_{i,\ell} = 1), \\ \frac{1}{2n(K-1)} & \text{otherwise (i.e., if } y_{i,\ell} = -1) \end{cases} \quad (2)$$

to create K well-balanced one-against-all classification problems. AdaBoost.MH builds the final classifier $\mathbf{f}$ as a sum of *base classifiers* $\mathbf{h}^{(t)} : \mathcal{X} \to \mathbb{R}^K$ returned by a *base learner* algorithm Base$(\mathbf{X}, \mathbf{Y}, \mathbf{W}^{(t)})$ for each iteration t. In general, the base learner should seek to minimize the *base objective*

$$E\big(\mathbf{h}, \mathbf{W}^{(t)}\big) = \sum_{i=1}^{n}\sum_{\ell=1}^{K} w_{i,\ell}^{(t)} \exp\big(-h_\ell(\mathbf{x}_i) y_{i,\ell}\big). \quad (3)$$

In our tests we used *discrete* AdaBoost.MH, in which the vector-valued base classifier $\mathbf{h}(\mathbf{x})$ is represented as

$$\mathbf{h}(\mathbf{x}) = \alpha \mathbf{v} \varphi(\mathbf{x}), \quad (4)$$

where $\alpha \in \mathbb{R}^+$ is the *base coefficient*, $\mathbf{v} \in \{+1, -1\}^K$ is the *vote vector*, and $\varphi(\mathbf{x}) : \mathbb{R}^d \to \{+1, -1\}$ is a *scalar* base classifier. It can be shown that to minimize (3), one has to choose a $\mathbf{v}$ and a φ that maximize the *edge*

$$\gamma = \sum_{i=1}^{n}\sum_{\ell=1}^{K} w_{i,\ell} v_\ell \varphi(\mathbf{x}_i) y_{i,\ell}. \quad (5)$$

[1] The indicator function $\mathbb{I}\{A\}$ is 1 if its argument A is true and 0 otherwise.

AdaBoost.MH.BA($\mathbf{X}, \mathbf{Y}, \mathbf{W}^{(1)}$, Base$(\cdot,\cdot,\cdot,\cdot), T, \mathcal{G} = \{\mathcal{H}_1, \ldots, \mathcal{H}_M\}$, BanditAlgo)

1 **for** $t \leftarrow 1$ **to** T

2 $\quad j \leftarrow$ BanditAlgo.$getArm()$

3 $\quad \mathbf{h}^{(t)}(\cdot) \leftarrow \alpha^{(t)}\mathbf{v}^{(t)}\varphi^{(t)}(\cdot) \leftarrow$ Base$\big(\mathbf{X}, \mathbf{Y}, \mathbf{W}^{(t)}, \mathcal{H}_j\big)$

4 $\quad \gamma_{\mathcal{H}_j}^{(t)} \leftarrow \sum_{i=1}^{n} w_i^{(t)} h^{(t)}(\mathbf{x}_i) y_i$ $\quad \triangleright$ *edge* $= 1 - 2 \times$ *error*

5 $\quad r_j^{(t)} = -\log\sqrt{1 - {\gamma_{\mathcal{H}_j}^{(t)}}^2}$ $\quad \triangleright$ *calculate reward based on edge*

6 $\quad$ BanditAlgo.$receiveReward(j, r_j^{(t)})$

7 $\quad$ **for** $i \leftarrow 1$ **to** n **for** $\ell \leftarrow 1$ **to** K

8 $$\quad w_{i,\ell}^{(t+1)} \leftarrow w_{i,\ell}^{(t)} \frac{\exp\big(-h_\ell^{(t)}(\mathbf{x}_i) y_{i,\ell}\big)}{\sum_{i'=1}^{n}\sum_{\ell'=1}^{K} w_{i',\ell'}^{(t)} e^{-h_{\ell'}^{(t)}(\mathbf{x}_{i'}) y_{i',\ell'}}}$$

9 **return** $\mathbf{f}^{(T)}(\cdot) = \sum_{t=1}^{T} \mathbf{h}^{(t)}(\cdot)$

Figure 2: The pseudocode of the bandit-accelerated version of AdaBoost.MH.BA algorithm.

The optimal coefficient is then

$$\alpha = \frac{1}{2}\log\frac{1+\gamma}{1-\gamma}.$$

The simplest scalar base learner used in practice is the *decision stump*, a one-decision two-leaf decision tree having the form

$$\varphi_{j,b}(\mathbf{x}) = \begin{cases} 1 & \text{if } x^{(j)} \geq b, \\ -1 & \text{otherwise,} \end{cases} \quad (6)$$

where j is the index of a feature and b is the decision threshold.

Although boosting decision stumps often yields satisfactory results, the state-of-the-art performance of AdaBoost is usually achieved by using *decision trees* as base learners, parametrized by their number of leaves N. We also tested a recently proposed base learner that seems to outperform boosted trees on large benchmarks [11]. This learner optimizes *products* of simple base learners of the form $\mathbf{h}(\cdot) = \alpha\prod_{j=1}^{m}\mathbf{v}_j\varphi_j(\cdot)$, where the vote vectors $\mathbf{v}_j$ are multiplied element-wise. The base learner is parametrized by the number of terms m.

2.2. Accelerated AdaBoost.MH

Running a full search for each boosting iteration step of AdaBoost.MH is prohibitively expensive, so we decided to run an accelerated version based on a multi-armed bandit (MAB) setup [12]. The general idea of accelerating the base learner is to partition the base classifier space $\mathcal{H}$ into (not necessarily disjunct) subsets $\mathcal{G} = \{\mathcal{H}_1, \ldots, \mathcal{H}_M\}$ and use MABs to learn the usefulness of the subsets. Each arm represents a subset, so, for each iteration, the bandit algorithm selects a subset. The base learner then finds the best base classifier *in the subset* (instead of searching through the *whole space* $\mathcal{H}$), and returns a reward based on this optimal base learner. Based on the rewards, the MAB algorithm learns the quality or usefulness of the subsets. Here, we used an adversarial bandit algorithm (EXP3.P [13]). An overview of AdaBoost.MH.BA is given in Figure 2.

For simple decision stumps, the most natural partitioning is to assign a subset to each feature: $\mathcal{H}_j = \{\varphi_{j,b}(\mathbf{x}) : b \in \mathbb{R}\}$.

Method		Acc.	UAR	UAAUC
n = 0	stumps	74.03%	76.60%	89.94%
	products	81.46%	77.45%	90.63%
	trees	82.98%	78.24%	91.34%
n = 4	stumps	76.12%	78.61%	91.03%
	products	81.88%	79.40%	91.79%
	trees	85.34%	80.71%	92.97%
n = 8	stumps	79.83%	81.23%	92.91%
	products	86.15%	81.93%	93.63%
	trees	87.73%	82.67%	94.26%
SVM (baseline)		70.00%	73.80%	87.60%

Table 1: *Results for the Social Signals Sub-Challenge.*

But the idea behind ADABOOST.MH.BA cannot be applied directly with decision trees or decision products, since it is hard to find a suitable partitioning of the space of trees or products, due to the size of this space. Thus, we followed the setup proposed in [14] in which trees and products are modeled as sequences of decisions over the smaller partitioning used for stumps.

We employed an open source implementation – the tool called `multiboost` [15] available at `http://multiboost.org` –, where this bandit-based approach is implemented along with the base learners mentioned above.

3. The Steps of Training

To get an optimal performance from a machine learning method on a specific learning task, we need to fine-tune the meta-parameters of the algorithm. Fortunately, the training data did not need preprocessing (e.g. feature selection or normalization), as AdaBoost was reported to be quite insensitive to the training parameters [12]. Hence, after handling some technical issues (removing all labels except one from the datasets and renaming the remaining one "class") we only had to find the best base learner; we tried decision stumps, products and decision trees.

3.1. Social Signals Sub-Challenge

This sub-challenge was quite similar to a standard speech recognition phoneme identification task, since it required frame-level classification with a relatively low number of frame-level features, suggesting that techniques developed for speech recognition could be successfully applied here as well. One such idea was that instead of classifying just one frame at a time, we should concatenate the features of the actual, the n left and the n right neighbouring frames into one large observation vector, while keeping the original class label. We made experiments with $n = 0$, $n = 4$ and $n = 8$. Since the "garbage" class was overrepresented, we reduced the number of examples of this class by sub-sampling: we retained just 85034 examples from this class in the train set (the same number as for the "filler" class). We did not downsize the development nor the test sets.

We performed only one training with $n = 0$, $n = 4$ and $n = 8$ for the decision stumps, products and decision trees, because our preliminary tests showed that, probably due to the large number of samples, there was only a slight difference in performance among models trained in the same way.

3.2. Emotion and Autism Sub-Challenges

Before training we had to remove the unnecessary examples from the emotion database (i.e. occurrences of class "other" for the Category task, and occurrences of class "unknown" for the Arousal and Valence tasks). In the Autism Sub-Challenge we performed two-step classification. First the typicality task was evaluated, then the examples considered atypical (class label "aty") were sorted into one of the three autistic category. 100 trainings were performed for each class with each base learner (decision stumps, products and decision trees). Then all the models were evaluated, stopping every 100 iterations, and the label of each example was determined by majority voting. The optimal iteration number was chosen as the one with the highest UAR score. Although this process does not result in the highest scores possible on the development set, we consider it a good method for obtaining stable training parameters.

Lastly, we chose the best-performing base learner and trained 100 models of it with the previously-obtained maximal iteration count on the joined train and development sets, and this model was submitted to the challenge.

Class	Set	Method	Acc.	UAR
Arousal	dev	stumps	83.17%	83.43%
		products	82.69%	82.85%
		trees	82.69%	82.85%
		SVM	82.20%	82.40%
	test	stumps	73.49%	73.87%
		SVM	—	75.00%
Valence	dev	stumps	80.77%	80.77%
		products	80.29%	80.29%
		trees	79.81%	79.81%
		SVM	77.88%	77.88%
	test	stumps	63.26%	63.26%
		SVM	—	61.60%
Category	dev	stumps	43.23%	45.69%
		products	40.10%	42.36%
		trees	39.58%	41.11%
		SVM	37.00%	40.10%
	test	stumps	41.87%	42.29%
		SVM	—	40.90%

Table 2: *Results for the Emotion Sub-Challenge.*

Class	Set	Method	Acc.	UAR
Typicality	dev	stumps	93.41%	93.69%
		products	93.04%	93.24%
		trees	93.16%	93.15%
		SVM	92.60%	92.80%
	test	stumps	93.17%	90.89%
		SVM	—	90.70%
Diagnosis	dev	stumps	76.92%	57.18%
		products	77.17%	57.92%
		trees	76.92%	57.18%
		SVM	69.80%	51.70%
	test	stumps	82.07%	62.10%
		SVM	—	67.10%

Table 3: *Results for the Autism sub-challenge.*

4. Results

After describing the techniques used for training, we will now present the results obtained. The results will be expressed in terms of accuracy score (*Acc.*) and Unweighted Average Recall (*UAR*); for the Social Signals Sub-Challenge we will also present the UAAUC score, which is the mean of the Area Under

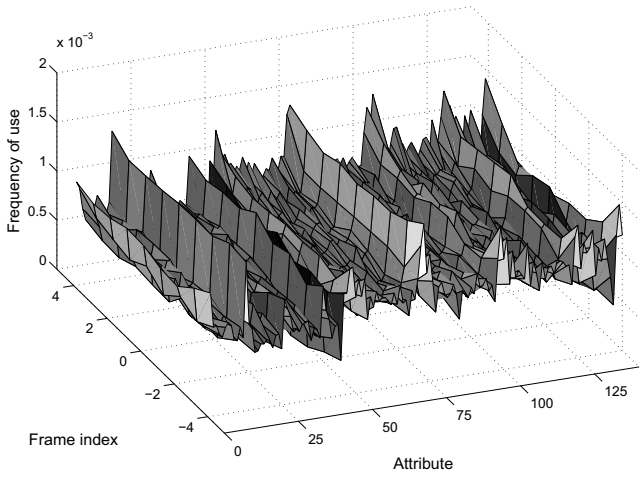

Figure 3: *Frequency of use of features with* $n = 4$.

Curve (*AUC*) value for the classes "filler" and "laughter". Besides the baseline Support Vector Machines (SVM) [16] score, we will give the scores for all three base learners got by voting on the development set, and the score got for the test set of the base learner which proved to be the best on the development set.

From the scores obtained for the Social Signals Sub-Challenge (see Table 1), it can be seen that the baseline SVM result was exceeded in the $n = 0$ case. But using the feature vectors of the neighbouring frames helps improve the classification even more: the UAAUC score of 94.26% for the case $n = 8$ and tree base learners is much higher than the baseline score of 87.60%. On the test set we achieved 89.70% for UAAUC, which is clearly superior to the SVM score of 83.30%.

The accuracy and UAR scores obtained for the Emotion sub-challenge (see Table 2) tell us that all base learners were able to outperform baseline SVM on all three tasks, except arousal on the test set. It is a bit surprising, that decision stumps proved to be the best, which may be due to the small number of examples. On the Autism sub-challenge (see Table 3) the results are acceptable; the reason for not being able to match the score of SVM on the test set is probably that, due to the relatively big size of development set, we failed to determine the optimal iteration number.

5. Using multiboost for Feature Selection

Since the *AdaBoost.MH.BA* algorithm gradually learns the usefulness of the features during the training process, it can be used for feature selection by examining how frequently it chose a particular feature during training. Next, we will examine the feature set from just this viewpoint.

5.1. Social Signals Sub-Challenge

In this challenge there were 141 attributes, growing to 1269 and 2397 for the $n = 4$ and $n = 8$ cases, respectively. Figure 3 shows how the frequency of use varied for each (original) attribute and relative frame index for $n = 4$. It is quite apparent that only a few features were frequently used, but it happened irrespective of the frame index. Among the 14 most frequently used features were all but one of those related to `pcm_LOGenergy` (8 attributes in total, the exception being the first derivative), the first MFCC coefficient, its mean and standard deviation, the means of its first and second order derivatives (5 attributes), and `pcm_zcr_amean`. Our findings for the $n = 8$ case were practically identical. This supports our hypothesis that for filler and laughter detection, we need mainly gross information about the spectral content of the current portion of speech and the fine spectral details (described, for example, by the remaining MFCC components) are less important.

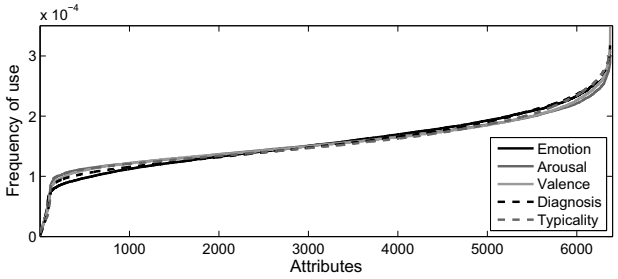

Figure 4: *Frequency of use of features for the tasks in the Emotion and Autism Sub-Challenges. Note that the frequencies are sorted separately for each task.*

Another observation is that for a given attribute the more distant frames were investigated more frequently than the closer ones (e.g. the features belonging to the earliest frame were used 20% more than the ones belonging to the middle frame for $n = 4$, and 30% more for $n = 8$). The reason for this is probably that the features already include some information about the neighbouring frames in the form of mean and standard deviation values on a 9-frame long sliding window; hence the feature vector positioned at 4 frames before the actual frame also contains information about the 8th preceding frame, while the values of the inner frames are half redundant.

5.2. Emotion and Autism Sub-Challenge

Figure 4 shows the distribution of attribute use frequency for the five tasks belonging to these two Sub-Challenges (note that the frequencies are sorted separately for each curve). Although the shape of the curves is practically identical, there is almost no overlap in the most frequently used attributes for the emotion sub-challenge (tasks emotion, arousal and valence). But there are a lot of common ones in the *least used* attributes: taking the 123 least important attributes for all three tasks, the list consists of only 175 separate features, 81 of them being practically ignored in all three tasks, most of them being some kind of minimum segment length ("`minSegLen`").

In the Autism Sub-Challenge, the 123-123 least used attributes take up only 132 separate items altogether. The most frequently used common attributes include `percentile1.0` of `audSpec_Rfilt_sma` 4, 5, 6, 20 and 21, and `lpcs` of `mfcc_sma` 4 and 5.

6. Conclusions

In the Computational Paralinguistics Challenge 2013, various tasks were set, where the goal was to retrieve paralinguistic information from speech recordings. Treating three sub-challenges as pure machine learning tasks, we applied AdaBoost.MH.BA to them. We outperformed the baseline SVM in almost every case, and in the Social Signals Sub-Challenge our results were excellent. Finally, we analyzed the feature set provided, based on the trained models.

7. References

[1] N. Morgan and H. Bourland, "An introduction to hybrid HMM/connectionist continuous speech recognition," *Signal Processing Magazine*, vol. May 1995, pp. 1025–1028, 1995.

[2] P. Bodnár and L. G. Nyúl, "Improving barcode detection with combination of simple detectors," in *International Conference on Signal Image Technology and Internet Based Systems (SITIS)*, Sorrento - Naples, Italy, 2012, pp. 300–306.

[3] D. A. van Leeuwen, A. F. Martin, M. A. Przybocki, and J. S. Bouten, "NIST and NFI-TNO evaluations of automatic speaker recognition," *Computer Speech & Language*, vol. 20, no. 23, pp. 128 – 158, 2006.

[4] L. Vidrascu and L. Devillers, "Detection of real-life emotions in call centers," in *Proceedings of Interspeech*, Lisboa, Portugal, 2005, pp. 1841–1844.

[5] S. L. Tóth, D. Sztahó, and K. Vicsi, "Speech emotion perception by human and machine," in *Proceedings of COST Action*, Patras, Greece, 2012, pp. 213–224.

[6] B. Schuller, S. Steidl, A. Batliner, A. Vinciarelli, K. Scherer, F. Ringeval, M. Chetouani, F. Weninger, F. Eyben, E. Marchi, H. Salamin, A. Polychroniou, F. Valente, and S. Kim, "The Interspeech 2013 Computational Paralinguistics Challenge: Social signals, Conflict, Emotion, Autism," in *Proceedings of Interspeech*, Lyon, France, 2013.

[7] R. A. Berk, *Statistical Learning from a Regression Perspective*, ser. Springer Series in Statistics. Springer, 2008.

[8] H. Cheng, N. Zheng, and C. Sun, "Boosted Gabor features applied to vehicle detection," in *International Conference on Pattern Recognition*, Hong Kong, 2006, pp. 662–666.

[9] M. Zhizhong, T. Huixin, and W. Yan, "Molten steel and temperature soft sensing in LF based on hybrid model using AdaBoost," *Chinese Journal of Scientific Instrument*, vol. 29, no. 3, pp. 662–667, 2008.

[10] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.

[11] B. Kégl and R. Busa-Fekete, "Boosting products of base classifiers," in *International Conference on Machine Learning*, vol. 26, Montreal, Canada, 2009, pp. 497–504.

[12] R. Busa-Fekete and B. Kégl, "Fast boosting using adversarial bandits," in *International Conference on Machine Learning*, vol. 27, 2010, pp. 143–150.

[13] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire, "The nonstochastic multi-armed bandit problem," *SIAM Journal on Computing*, vol. 32, no. 1, pp. 48–77, 2002.

[14] R. Busa-Fekete and B. Kégl, "Accelerating AdaBoost using UCB," in *KDDCup 2009 (JMLR W&CP)*, vol. 7, Paris, France, 2009, pp. 111–122.

[15] D. Benbouzid, R. Busa-Fekete, N. Casagrande, F.-D. Collin, and B. Kégl, "MultiBoost: a multi-purpose boosting package," *Journal of Machine Learning Research*, vol. 13, pp. 549–553, 2012.

[16] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.