Ensemble of Machine Learning Algorithms for Cognitive and Physical Speaker Load Detection

How Jing[1], *Ting-Yao Hu*[2], *Hung-Shin Lee*[3], *Wei-Chen Chen*[4], *Chi-Chun Lee*[4], *Yu Tsao*[1], *Hsin-Min Wang*[3]

[1]Research Center for Information Technology Innovation, Academia Sinica
[2]Graduate Institute of Communication Engineering, National Taiwan University
[3]Institute of Information Science, Academia Sinica
[4]Department of Electrical Engineering, National Tsing Hua University

jingt1228@gmail.com, benken.tyhu@gmail.com, hungshinlee@gmail.com, cclee@ee.nthu.edu.tw

Abstract

We present our methods and results on participating in the Interspeech 2014 Computational Paralinguistics ChallengE (ComParE) of which the goal is to detect certain type of load of a speaker using acoustic features. There are in total seven classification models contributing to our final prediction, namely, *neural network with rectified linear unit and dropout* (ReLUNet), *conditional restricted Boltzmann machine* (CRBM), *logistic regression* (LR), *support vector machine* (SVM), *Gaussian discriminant analysis* (GDA), *k-nearest neighbors* (KNN), and *random forest* (RF). When linearly blending the predictions of these models, we are able to get significant improvements over the challenge baseline.

Index Terms: Physical Load Detection, Cognitive Load Detection, Neural Network, Classification Models

1. Introduction

Paralinguistic analysis such as the recognition of speech emotion and pathology is increasingly turning into a mainstream topic in speech and language processing. Automatic detection of certain types of physical/mental or cognitive states of humans from the speech is becoming much more important. This paper reports the results and the methodology of the two subchallenges in the INTERSPEECH 2014 Computational Paralinguistics Challenge (ComParE) [1]. In *physical load subchallenge*, given an input speech, we are to classify his/her exercising state as either *running* or *resting* and by that the heart rate as either *high pulse* or *low pulse*. In the *cognitive load subchallenge*, we are to classify speaker's cognitive load state as either *low*, *medium* or *high*, based on the acoustic features.

In our system, we use the comprehensive features in the standard datasets, which is a set of 6373 dimensional vectors, and we build seven off-the-shelf classification models, of which some are well-known to the machine learning society for a long time, and some are recently developed techniques. The seven classifiers we use are: *neural network with rectified linear unit and dropout* (ReLUNet), *conditional restricted Boltzmann machine* (CRBM), *logistic regression* (LR), *support vector machine* (SVM), *Gaussian discriminant analysis* (GDA), *k-nearest neighbors* (KNN), and *random forest* (RF). Training in our system consists of two stages. First, each model is trained individually to fit the training data with the hyperparameters for each model optimized via investigating the performance on the validation set. In the second stage, we ensemble these models linearly by training another logistic regression model that takes the outputs given by the seven models as its input, and the weight vector in this logistic regression model determines the relative importance of each model that contributes to the final predictions. The parameters in the logistic regression in the second stage are learned using the validation set to avoid severe overfitting. At the testing phase, we simply predict the output by an weighted average of individual predictions, where the weights associated to each model's individual prediction is determined via the logistic regression model trained in the second stage. Figure 1 shows the overall training procedure of our system.

The remaining sections are organized as follows: Section 2 briefly reviews five out of seven classifiers that are well-known to the machine learning community, namely, SVM, LR, KNN, GDA and RF. Section 3 gives details about two somewhat recently developed techniques, called the conditional restricted Boltzmann machine (CRBM) and the neural network along with rectified linear units and dropout (ReLUNet). Section 4 provides description about how we ensemble the predictions of 7 models. Section 5 reports the experimental results on the validation set and compares to the challenge baseline. Section 6 concludes this paper with a short summary.

From this point, we assume that we are given a training set $\{(x_i^{tr}, y_i^{tr})|i = 1 : N^{tr}\}$, a validation set $\{(x_i^{val}, y_i^{val})|i = 1 : N^{val}\}$ and a unlabeled test set $\{(x_i^{te})|i = 1 : N^{te}\}$, where $x \in \mathcal{R}^D$.

2. Off-The-Shelf Classifiers

2.1. Support Vector Machines and Logistic Regression

Support Vector Machine (SVM) is a popular classification method that has been extensively used in many applications [3]. It offers promising results especially for a binary classification problem. In particular, given a set of training data, it seeks a weight vector w that maximizes the 'margin' by solving the following optimization problem

$$\min_w \frac{1}{2}||w||_2^2 + C\sum_{i=1}^{N^{tr}} \epsilon_i$$
$$s.t.\ \ y_i(w^T x_i + b) \geq 1 - \epsilon_i$$
$$\epsilon_i \geq 0, i = 1, ..., N^{tr}$$

where C is the regularization parameter that allows the optimization to work for a non-separable dataset and to prevent overfitting. The core computation involved in SVM is a convex quadratic programming problem. Usually, an efficient Sequen-

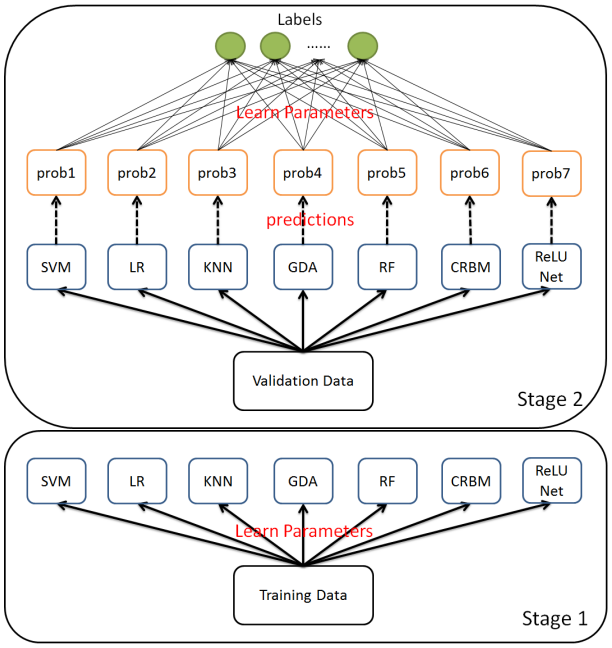

Figure 1: Training in our system consists of two stages. In the first stage, we learn parameters of individual classifiers separately to fit the training data. In the second stage, we learn the parameters of another logistic regression model that is responsible for blending the predictions of each classifier.

tial Minimal Optimization (SMO) algorithm is used in training SVM [3]. For multi-class tasks, as in the *cognitive load sub-challenge*, we adopt the common one-vs-all approach. The balancing problem could be an issue in training because SVM optimizes the global training accuracy and may bias towards classes that have more training instances. This issue is explicitly pointed out in the challenge baseline documentation for the *cognitive load sub-challenge*, and we follow the approach used in the baseline by doing a down-sampling step before training the SVM.

Logistic Regression (LR) is also a very popular classification model. It aims at maximizing the conditional log-likelihood

$$\sum_{i=1}^{N^{tr}} \log p(y_i^{tr}|x_i^{tr}) = \sum_{i=1}^{N^{tr}} \log \frac{1}{1+\exp(-w^T x_i^{tr} - b)}$$

with respect to a weight vector w and a bias parameter b. LR can be generalized to multi-class problems via the *softmax* function. Both SVM and LR involve solving a convex optimization objective and hence global optimum can be found.

2.2. K Nearest Neighbors

K-Nearest Neighbors (KNN), also called the lazy learning algorithm, does not have any training. It makes predictions 'on-the-fly' by retrieving similar data from the training set, and determines the class label of a test query by combining information from its retrieved neighbors. The simplest metric for retrieving neighbors is the euclidean distance, and the simplest classification rule is a simple majority vote,

$$vote(y_i^l|x^{te}) = \sum_{\{i|x_i \in \mathcal{N}(x^{te})\}} y_i^l$$

$$pred(x^{te}) = \arg\max_l vote(y_i^l|x^{te})$$

Variants of KNN are used widely in various domains [13, 2]. In our system, we use the above simplest version.

2.3. Gaussian Discriminant Analysis

Gaussian Discriminant Analysis (GDA) is another linear model that treats the class conditional densities as Gaussian distributed. The model fits a Gaussian for each class and assumes that all classes share the same covariance matrix. Specifically, for each class it fits a Gaussian density model by learning its mean and covariance matrix. Using Bayes rule we have the following optimization problem for each class c,

$$\arg\max_{\mu_c,\Sigma} \prod_{y_i=c} p(y_i^c = 1|x_i) =$$

$$\arg\max_{\mu_c,\Sigma} \sum_{y_i=c} \log p(x_i|y_i^c) + \log p(y_i^c)$$

where $p(x|y_i^c) \sim \mathcal{N}(\mu_c, \Sigma)$ is the conditional Gaussian and the $p(y_i^c)$ is the class prior for the class c. Using maximum a posteriori estimation, we could find the parameters in this model. At the testing phase, it computes the posterior probability for each class and selects the one that gives the highest probability.

2.4. Random Forest

Random Forest (RF), an ensemble model that averages the predictions of many decision trees, can give good generalization ability on classification/regression problems [9]. It learns several decision trees where each tree takes a different subset of the training data. Presumably, each tree will learn very different decision boundary hence when we average thousands of their predictions, we could get much higher performance than using a single decision tree. For more information about RF, we refer to [9].

3. CRBM and ReLUNet

3.1. Conditional Restricted Boltzmann Machines

Conditional Restricted Boltzmann Machines (CRBMs) are undirected models that learns the conditional distribution $p(y|x)$. The following energy function is defined,

$$E(x,h,y) = -x^T W h - y^T W h - c^T h - d^T y$$

where h and y are binary stochastic units that can take on values $\{0,1\}$, and the input x provides bias to the hidden units. $\{W,U\}$ are weight matrices that capture the pair-wise interactions between (x,h) and (y,h) respectively, and c,d are bias terms for h and y. The conditional probability distribution is thus defined by,

$$p(y|x) = \frac{\sum_h \exp(-E(x,h,y))}{\sum_{h,y} \exp(-E(x,h,y))}$$

Figure 2 shows the graphical representation of a CRBM. This model can be viewed as using a single stochastic hidden layer h to capture the correlations between input and output. Unlike

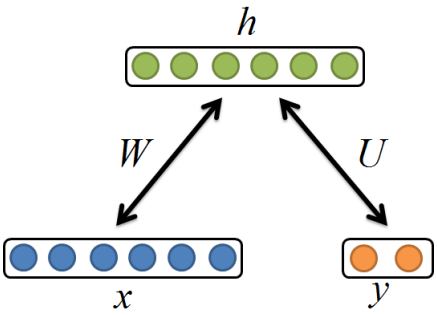

Figure 2: A conditional restricted Boltzmann machine (CRBM). The hidden layer captures the correlations between visible input x and the label vector y.

the traditional restricted Boltzmann machine (RBM), CRBM is usually not used for estimating the density of the data. Instead, such model (and its variants) has been used extensively in discriminative learning tasks such automatic music tagging [10], modelling the motion style [12]. Learning in a CRBM can be done using gradient ascent in the log-likelihood, where the gradient takes the following form (for some parameter $\theta \in \{W, U, c, d\}$),

$$\frac{\partial \log p(y|x)}{\partial \theta} = \sum_h p(h|x,y)\frac{\partial - E(x,h,y)}{\partial \theta} - \sum_{h,y} p(h,y|x)\frac{\partial - E(x,h,y)}{\partial \theta}$$

where the first term is also referred to the data-dependent expectation, and the second is the model-dependent expectation. Due to the special structure of this graphical model, the data-dependent expectation can be computed exactly in time linear to the number of hidden units. However, the second term is computationally intractable because there are exponentially many configurations to be visited. It is possible to define a Markov Chain Monte Carlo (MCMC) procedure that attempts to draw samples from the model distribution. Specifically, block Gibbs sampling procedure can be defined by alternating between sampling h and sampling y. However, since we have to run such procedure until convergence for every data point in every gradient update, it is very slow in practice. Nonetheless, if we just run a few steps of Gibbs sampling and take the 'truncated' samples anyway to update the parameters, it is very efficient. This algorithm is called the contrastive divergence (CD) algorithm, and it requires very little modification of the full MCMC algorithm [7]. It clearly does not follow the correct maximum likelihood gradient, but it works well in practice.

3.2. Neural Networks with ReLU and Dropout

Deep neural networks (DNN) have shown promising results in speech recognition and classification in the past few years [14, 4, 5]. Two major techniques proposed recently have improved the performance of DNN significantly. First, a different type of neurons called the rectified linear units (ReLU) works much better than conventionally used logistic units for several tasks. Mathematically, a feed-forward neural network consists of multiple layers of hidden neurons, where each neuron is activated via an activation function $f(s)$ and s is the total input it receives. Typical choices of activation functions are logistic function: $\frac{1}{1+\exp(-s)}$ and hyperbolic tangent: $\frac{e^s - e^{-s}}{e^s + e^{-s}}$. ReLUNet replaces the activation function by $max(0, s)$. The advantage of ReLU is that it generalizes better than logistic units and empirical results also show that it converges faster [11].

The second technique is called the dropout [11, 8]. Dropout can be viewed as a strong regularization technique that prevents the co-adaptation of hidden units, so that each hidden unit needs to learn useful patterns in the data individually. It can also be viewed as an approximation of an ensemble method that takes the geometric average of exponentially many neural networks, each with different architecture. Usually, it is easy for neural network to overfit when the number of labelled training data is scarce. Dropout allows us to use a big deep net and regularize it without paying much cost.

4. Models Ensemble

Inevitably, the seven classifiers we use all overfit the data because the number of training data is very scarce. To eliminate this problem, we need to average predictions made by different models. Presumably, each model can output a probability distribution over the classes which is treated as the confidence of each model. We could then blend these results by learning another set of weights to decide the relative importance of each model given a test vector. Specifically, let p_i be the predictive distribution vector formed by concatenating the predictions of the seven models we have learned, we train another LR,

$$\arg \max_{w_{ensemble}} \sum_{i=1}^{N^{val}} \log p(y_i^{val}|p_i^{val})$$

i.e., we use the predictions of the seven classifiers as our new data, and train the ensemble weight to maximize the probability of the correct predictions on the **validation set**. It is important that we do not use the training set to learn these ensemble weights because there is severe issue of overfitting and nearly every classifier gets very high accuracy on the training set.

This process of learning the ensemble weights is referred as the second stage of our system. At testing time, we get the predictions of the seven classifiers, and treat those as data and feed them into the model in the second stage to get the final output.

5. Experiments

5.1. Dataset and Evaluation

The datasets for the two sub-challenges can be downloaded from the competition website [1], where the baseline features, extracted by using the openSMILE feature extractor are provided [6]. The features lie in 6373 dimensional space for both sub-challenges. As mentioned in Section 2.1, in the *cognitive load sub-challenge*, we down-sample instances tagged as *high pulse* to make the training data more balanced.

D	L	N^{tr}	N^{val}	N^{te}
256	2	385	384	319

Table 1: Statistics of the *physical load sub-challenge*. D is the feature dimension, L is the number of classes.

The evaluation metric used in the competition is the *unweighted average recall* (UAR). UAR is calculated by computing the recall rate for each class individually, and compute their

[1] http://emotion-research.net/sigs/speech-sig/is14-compare

D	L	N^{tr}	N^{val}	N^{te}
400	3	1023	651	744

Table 2: Statistics of the *cognitive load sub-challenge*. D is the feature dimension, L is the number of classes.

unweighted average. We follow this metric and the parameters tuned on the validation set are all based on optimizing UAR.

5.2. Pre-processing

In our system, we did not use features other than the baseline ones. However, to speed up computation and robustify the classifiers, we reduce its dimensionality by performing a PCA first. We select the top-K principal components with the largest variances by investigating the performance achieved by SVMs of the reduced features on the validation set. We found that for K ranging between 200 and 500, the performance of SVMs are similar to and sometimes even better than the results reported in the baseline documentation. The final number of K is set to 256 for *physical sub-challenge* and 400 for *cognitive sub-challenge*. We list some statistics about the (pre-processed) datasets in Table 1 and 2.

Note that for correct evaluation of the classifiers, PCA is performed on the training set only. We do not want to overestimate the performance of each classifier by doing PCA on both the training set and the validation set. Although, as pointed out in the baseline paper, that the full test set (without labels) is given to the participants and it should be okay to pre-process features with all available data at hand. However, we focus on a more reasonable scenario where the test query is coming to us in an online fashion that we are not allowed to perform any kind of learning using the test data.

5.3. Results

We now report the results of the seven classifiers as well as the final ensemble model we use in our system on both sub-challenges. For each classifier, we try several hyperparameters from a predefined range of choices, e.g., the regularization parameter C for SVM and LR, the number of neighbors for KNN, and the number of decision trees for RF. For CRBM, we choose the number of hidden units beforehand. For ReLUNet, we try 1 to 3 hidden layers, but we found out that using more layers does not give substantial performance improvements, probably because that the training data is too scarce, so using even 1 hidden layer suffices to capture enough information in the data.

5.3.1. Physical sub-challenge

We first show the results on *physical sub-challenge*. In this task, the baseline model, which normalize the features to zero-mean and unit variance, and learns a linear SVM on it gets the UAR of **65.79**. The results of different classifiers are shown in Table 3. Due to the space limitation, we only show the hyperparameter setting that achieves the best result on the validation set for each model.

5.3.2. Cognitive sub-challenge

Subsequently, we report the results on the *cognitive sub-challenge*. There is one difference for this sub-challenge from the *physical load sub-challenge* that it is actually a *multi-task* problem. There are three tasks that each speaker has to perform, namely, *readingspanSentence*, *strooptimepressure*, and *stroop-dualtask*, all of which we need to predict speaker's cognitive load. We could learn separate model for each task, or we could learn only one model to solve three tasks jointly. Here we take the former approach, because it was reported in the baseline that learning each model for each task performs much better than learning a single classifier. Similarly, the results are shown in Table 4.

Model	UAR
baseline	65.79
SVM, *C*=0.5	68.22
LR , *C*=0.5	68.44
KNN, *NN*=20	60.66
RF, *NT* = 10000	68.04
CRBM, *H* = 100	69.20
ReLUNet,*H* = 500	67.66
GDA	65.64
ensemble	**71.40**

Table 3: UAR of SVM, LR, KNN, RF, CRBM, ReLUNet, GDA and the ensemble model *physical sub-challenge*. *C* is the regularization parameter for LR and SVM. *NN* is the number of neighbors in the KNN classifier, *NT* is the number of trees in RF and H is the number of hidden units in CRBM and ReLUNet.

Model	UAR
baseline	62.26
SVM, *C*=0.0001	62.10
LR , *C*=0.001	61.87
KNN, *NN*=10	52.80
RF, *NT* = 500	53.92
CRBM, *H* = 1000	58.64
ReLUNet,*H* = 100	61.93
GDA	56.03
ensemble	**64.05**

Table 4: UAR of SVM, LR, KNN, RF, CRBM, ReLUNet, GDA and the ensemble model for *cognitive sub-challenge*

Finally, we upload the results for the test set to the judge system. Unfortunately, we found that our results are slightly worse than the baseline method and we believe that the reason is we are overfitting the validation set too much.

6. Conclusion

We have presented our method and results on the Interspeech 2014 Computational Paralinguistics ChallengE (ComParE). We blend seven off-the-shelf classifiers for a better generalization power. As shown in the experiments, we are able to significantly improve upon the baseline model with our system. There are however, several interesting future directions worth working on. First, we did not do anything special on the feature engineering part. We used the openSMILE features and reduce its dimensionality by PCA. Given that the task is load detection, there should be special kind of features that are particularly suitable for such problems. Second, it is possible to perform a multi-modal learning. For example, in the *cognitive load sub-challenge*, the goal is to predict the *objective load* of the speakers. However, the *subjective load*

7. References

[1] Anton Batliner Julien Epps Florian Eyben Fabien Ringeval Erik Marchi Yue Zhang Bjrn Schuller, Stefan Steidl. The interspeech 2014 computational paralinguistics challenge: Cognitive & physical load. In *Proceedings INTERSPEECH 2014*, 2014.

[2] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008.

[3] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[4] George E. Dahl, Tara N. Sainath, and Geoffrey E. Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *ICASSP*, pages 8609–8613. IEEE, 2013.

[5] Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *in Proc. Int. Conf. Acoust., Speech, Signal Process*, 2013.

[6] Florian Eyben, Martin Wöllmer, and Björn Schuller. Opensmile: The munich versatile and fast open-source audio feature extractor. In *Proceedings of the International Conference on Multimedia*, MM '10, pages 1459–1462, 2010.

[7] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation.*, 14:1771–1800, 2002.

[8] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[9] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2:18–22, 2002.

[10] Michael Mandel, Razvan Pascanu, Hugo Larochelle, and Yoshua Bengio. Autotagging music with conditional restricted boltzmann machines. 2011.

[11] Nitish Srivastava. Improving Neural Networks with Dropout. Master's thesis, University of Toronto, Toronto, Canada, January 2013.

[12] Graham W. Taylor and Geoffrey E. Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1025–1032, 2009.

[13] Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research.*, 10:207–244, 2009.

[14] M.D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q.V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G.E. Hinton. On rectified linear units for speech processing. In *38th International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.