LEARNING DYNAMIC FEATURES WITH NEURAL NETWORKS FOR PHONEME RECOGNITION

Xin Zheng[⋆†], *Zhiyong Wu*[⋆†‡], *Helen Meng*[⋆‡] *and Lianhong Cai*[⋆†]

[⋆] Tsinghua-CUHK Joint Research Center for Media Sciences, Technologies and Systems
Shenzhen Key Laboratory of Information Science and Technology
Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China
[†] Tsinghua National Laboratory for Information Science and Technology (TNList)
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[‡] Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong SAR, China

zhengx11@mails.tsinghua.edu.cn, zywu@sz.tsinghua.edu.cn,
hmmeng@se.cuhk.edu.hk, clh-dcs@tsinghua.edu.cn

ABSTRACT

Dynamic features such as delta and delta-delta of basic acoustic features have long been used in various speech applications and give satisfactory performance. The explicit physical meaning and simplicity of dynamic features clearly compound their prevalence. In this paper, we propose a new framework with neural network to learn the alternatives of traditional delta and higher order differences. Instead of embracing the interpretability and simplicity, our framework is able to learn a new transformation that simulates what differences do but is more relevant to a specific task such as phoneme recognition. We determine the best way to learn such a new transformation among several most probable alternatives. Our experiments indicate that dynamic features obtained with transformation learned this way are better than traditional differences in both frame classification and phoneme recognition. The improvement of performance is even clearer when higher-order of differences are applied.

Index Terms— phoneme recognition, neural network, deep neural network (DNN), delta, higher order, difference

1. INTRODUCTION

Traditional dynamic features like delta and delta-delta of cepstrum coefficients have been shown to be very helpful in hidden Markov model (HMM)-based continuous speech recognition [1][2][3]. Traditional delta and delta-delta features have a clear physical meaning (velocity and acceleration), and the way to calculate them is very simple. Thus acoustic feature such as Mel-frequency cepstral coefficients (MFCC) and perceptual linear prediction (PLP) concatenated with delta and delta-delta have become the default way to use feature in building speech recognition systems.

Although delta and delta-delta features are useful, the simple calculation may have limited their power. To deal with such problem, Chengalvarayan and Deng [4] proposed generalized dynamic feature parameters in HMM-Gaussian mixture model (GMM) speech recognition system for phoneme recognition. Generalized dynamic feature parameters extend the traditional way to calculate delta features to a weighted linear combination of a window of several frames, with the weights to be learned. Their results show that dynamic features calculated with generalized dynamic feature parameters are much better than traditional dynamic features on TIMIT.

In recent years, deep learning techniques have attracted attention from speech community due to impressive performance in speech recognition. Researchers have found that replacing Gaussian mixture model (GMM) with deep neural network (DNN) can significantly improve the recognition accuracy of both phoneme recognition [5] and large vocabulary continuous speech recognition (LVCSR) tasks [6][7]. However, all the experiments in these results use raw acoustic features concatenated with traditional delta and delta-delta features. Deng et al. [8] performed the experiment of using only raw Mel-scale log filter bank with hybrid HMM-DNN, but got poor results. They thus concluded that DNN is not capable of capturing the concept of traditional delta and delta-delta features. However, no further discussion on whether different architectures can make a difference in learning these concepts was given.

In this work, we propose a framework with neural network to learn alternatives of traditional delta and delta-delta features. By pointing out the relationship between the calculation of delta features and the operation of one layer neural network, we design architectures to simulate the behavior of traditional dynamic features, yet providing a much more generalized possibility of building the transformation. With back-propagation algorithm, the transformations to build our dynamic features can be learned by optimizing the cross-entropy error of training data. Compared with Chengalvarayan and Deng's work [4], we are able to leverage the information of cross-coefficients transformation.

2. RELATED WORK

Traditional delta and delta-delta features are calculated as follows:

$$d_t = \frac{\sum_{\theta=1}^{\Theta} \theta \cdot (c_{t+\theta} - c_{t-\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2} \quad (1)$$

$$dd_t = \frac{\sum_{\theta=1}^{\Theta} \theta \cdot (d_{t+\theta} - d_{t-\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2} \quad (2)$$

in [9], where c_t, d_t, dd_t are static, delta and delta-delta coefficients at time t respectively. Θ is called delta window in HTK[9]. Higher-order differences can be similarly derived.

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

Besides speech recognition, formulation (1) and (2) have also been utilized to various of other speech applications such as speaker recognition [10], speech synthesis [11] and articulatory inversion [12]. While the physical meaning underlying these formulations has been shown to contain certain discriminative power, it is quite unreasonable that such formulations are always the only and best choice to calculate dynamic features from application to application. Instead of embracing such generality and interpretability, we would like to learn dynamic features that have the largest discriminative power for specific applications such as phoneme recognition.

3. LEARNING FRAMEWORK

3.1. Description

One of the most noteworthy thing in formulation (1) is that d_t is a linear transformation (combination) of $c_{t+\Theta}$ to $c_{t-\Theta}$. So if we are to design an architecture that is able to imitate the way delta coefficients behave, we would like it to be able to perform transformation f to static coefficients. That is, we would like to learn f that satisfies the following formulation:

$$d_t = f(c_{t-\Theta}, c_{t-\Theta+1}, \ldots, c_{t+\Theta-1}, c_{t+\Theta}) \tag{3}$$

such f can either be linear or non-linear. Note that such a transformation is easily implemented by a single layer of a neural network. Concretely, assuming both c_t and d_t have dimension $d \times 1$, $\Theta = 1$, with a weight matrix W of dimension $d \times 2d$ and a bias vector b of dimension $d \times 1$, we have:

$$d_t = g(W[c_{t-1}; c_{t+1}] + b) \tag{4}$$

where the operation $[p; q]$ represents stack vector p and q vertically and g can be identity function if we want linear transformation and sigmoid or tanh function if we want non-linear one. Although such a transformation constructed can not represent every possible function, it is capable of recovering formulation (1) by setting:

$$W = \begin{pmatrix} -0.5 & 0 & 0.5 & 0 \\ 0 & -0.5 & 0 & 0.5 \end{pmatrix}, b = \mathbf{0} \tag{5}$$

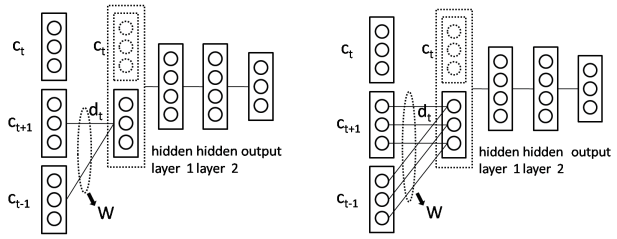
if $d = 2$ and $\Theta = 1$. Thus formulation (1) is just a special case of formulation (4). Also, if we would like to make even more complex transformations, we can simply add another layer on top of formulation (4). However, in some of our preliminary experiments, we found out that a single layer is enough to exceed the traditional delta formulation, and adding one more layer does not help.

After making the calculation of delta-like features under the framework of neural network, the rest part is the same way with traditional 39-dimensional MFCC used in a DNN. Thus, the training of the whole neural network is modifying not only the part that an ordinary neural network does, but also the weights that calculate delta-like features simultaneously. Since the weights that calculate delta-like features are a much more general case, they can be counted on to learn a better transformation that is relevant to the certain task the neural network optimizes. For example, in the phoneme recognition problem, the minimization of cross-entropy error provides a transformation to calculate delta-like features, together with basic acoustic features, minimizes the cross-entropy of the training data in the neural network.

3.2. Architectures

3.2.1. Learning new delta features

The idea described above can be illustrated in figure 1(a). All the solid lines represent a full connection and all biases are omitted. The units with dashed line of c_t are actually the same with solid line units c_t, and they are drawn this way only to make a convenient comparison between our framework and the ordinary way delta feature is used, by which we mean that the ordinary way to use delta feature does not contain the part on the left of the large dashed rectangle. Also, the two hidden layers can be extended to any number of hidden layers appropriate. The most important thing in this architecture is that all the solid lines between group of units in figure 1 are trained together. With back-propagation algorithm minimizing cross-entropy error, when the neural network converges, the weight matrix W in figure1, together with the omitted bias vector, creates an alternative transformation that is originally calculated with formulation (1). For convenience, we would like to call the transformation obtained this way the "new delta" in the following text. We will compare the original one and the new one later in section 4.4.



(a) full connection (b) sparse connection

Fig. 1. Two different architectures to learn delta-like features.

3.2.2. Learning new delta-delta features

While it is relatively easy to determine the architecture to learn delta-like features, the delta-delta-like features need some more discussion. Although delta-delta features are calculated from delta features by replacing d_t and c_t in formulation (1) to dd_t and d_t respectively, we point out that dd_t is still a linear transformation of $c_{t-2\Theta}$ to $c_{t+2\Theta}$ by substituting (1) into (2). So, as formulated in (3), we have:

$$dd_t = f(c_{t-2\Theta}, c_{t-2\Theta+1}, \ldots, c_{t+2\Theta-1}, c_{t+2\Theta}) \tag{6}$$

Following the discussion of section 3.1, the architecture in our framework can be illustrated figure 2(a) when $\Theta = 1$. Of course, delta-delta-like features can also be obtained from transformation of delta-like features, that is,

$$dd_t = f(d_{t-\Theta}, d_{t-\Theta+1}, \ldots, d_{t+\Theta-1}, d_{t+\Theta}) \tag{7}$$

and the architecture is illustrated in figure 2(b). Similarly, we would like to call the transformation obtained by both architectures the "new delta-delta". We will find out which of these two architectures in figure 2 is better later in section 4.4.

3.2.3. Learning higher-order differences

Following the discussion in section 3.2.2, it is quite natural to generalize the transformation to even higher-order differences beyond

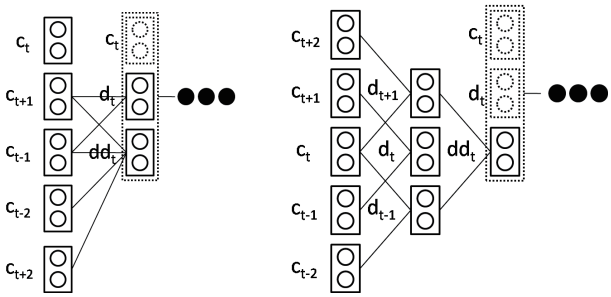


Fig. 2. Two different architecture to learn delta-delta-like features. The black dots represent the omitted two hidden layers and the output layer as illustrated in figure 1.

the 2nd order. Although they are infrequently used in speech recognition, higher order of differences in our architecture possess much more variability over traditional way of calculating differences than the lower order one, thus potentially imply better performance. We will report our experimental results later in section 4.5.

3.2.4. Full connection vs. sparse connection

In all the above architectures, weights are all fully connected, which means all elements of W in (4) are tunable. But the original way how delta feature is calculated only to transform feature in each dimension separately. In this case, similar to (5), we have:

$$W = \begin{pmatrix} x_1 & 0 & x_3 & 0 \\ 0 & x_2 & 0 & x_4 \end{pmatrix} \tag{8}$$

where x_i are numbers to learn. Such kind of transformation, if represented in our framework, can be illustrated in figure 1(b). To emphasize the difference, we call such kind of connections "sparse connections". The most essential question for the choosing between full connections versus sparse connections is whether cross-coefficients transformation is able to contribute to the classification. We will compare them later in section 4.3.

4. EXPERIMENTS

4.1. Corpus

We conducted all our experiments on TIMIT. MFCC features are extracted with a window size of 25ms and a window shift of 10ms. We used 12 cepstrum coefficients with the 0'th (13 dimensions) for basic MFCC and a delta window $\Theta = 2$ to calculate all the differences.

4.2. Setup

The first part of our experiments tries to find out the most appropriate way to learn new delta features from a few probable candidates. The second part compares the new delta and new delta-delta with the traditional ones. The last part tests the performance of the learned new different order of differences in phoneme recognition tasks. In all three parts of our experiments, data used to train a neural network are normalized to zero mean and unit variance in each dimension, in order to achieve faster and better convergence.

In the first two parts of our experiments, we used the same setting for training: 2 hidden layers (if not explicitly mentioned otherwise), each with 500 units; stochastic gradient descent (SGD) with a mini-batch size of 100; a constant learning rate 0.1 (apply to each mini-batch). We did not use momentum in these parts of our experiments, because employing momentum is very tricky and may significantly effect the final results [13] while in these experiments we just intended to make a fair comparison. Since there are 61 phonemes in TIMIT and we used a 3-states HMM for each phoneme, there are 183 classes altogether. All the labels of frames are obtained by an alignment from a HMM-GMM system. We used cross-entropy as training criterion. All the results in these two parts of our experiments are reported on the development set.

In the third part of our experiments, we used the type of transformation and configuration that were concluded from and described in the first two parts of our experiments to trained neural networks to learn different order of new differences. With these new differences, acoustic features of a context of 15 frames are put into a hybrid HMM-DNN system for phoneme recognition. Of all the experiments in this part, we used a feed-forward neural network of 6 hidden layers, each with 2000 units. All the neural networks are pretrained with stacked restricted Boltzmann machine the same way as in [14]. Cross-entropy was minimized with SGD with momentum. All the other hyper-parameters are the same as used in [14]. The recognition results are folded from 61 phones to 39 phonemes the same way as in [15].

4.3. Learn new delta features

In the first part of our experiments, we would like to determine which way is better in our framework to learn new delta features. Following the discussion in section 3.2.4, we compared the performance of full-connected and sparse-connected way of learning new deltas. The architectures have been illustrated in figure 1.

When we regard the layer that transforms basic MFCC to delta-like features as a hidden layer of the whole neural network, it is plausible that a non-linear activation function such as sigmoid function gives better results. However, the traditional way like (1) clearly shows an empirical linear activation function. So it would be interesting to find out which activation function is better.

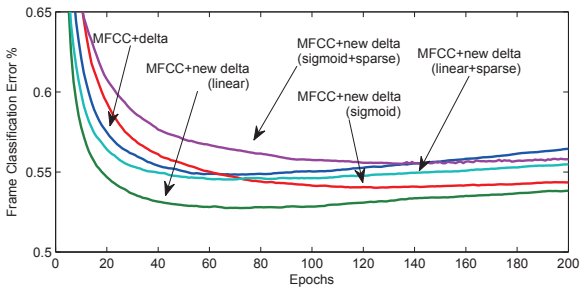


Fig. 3. Frame classification error rate on the development set of TIMIT by comparing sigmoid and linear activation function, as well as full and sparse connection.

Figure 3 shows the frame classification error of different setting on the development set. The first thing we can observe is that all sigmoid versions are worse than their linear counterparts, which validates the efficacy of the traditional way of calculating delta. The next thing worth noting is the comparison between the traditional

version of delta and the linear+sparse version. Our results show that a more general weight matrix like (8) is better than (5). Finally, we can see that a full-connected weight matrix W with linear activation gives best result, which we believe shows that cross-coefficients transformation is important for the classification of MFCC.

4.4. Compare with the traditional delta and delta-delta

According to section 4.3, we used fully connected weight matrix with linear activation for calculating all the new delta and new delta-delta in this section. Figure 4 compares the performance of traditional MFCC (with differences) with the ones we proposed, where type1 and type2 denotes the two different architectures of calculating new acceleration we presented in section 3.2.2 respectively (figure 2).

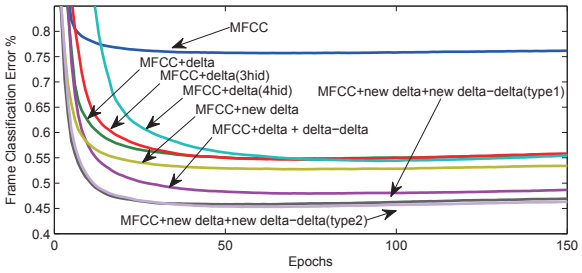


Fig. 4. Frame classification error rate on the development set of TIMIT by comparing traditional delta and delta-delta features and the new ones learned from our framework.

The results show that learning new delta-delta from new delta is slightly better than from MFCC, which means that the architecture in figure 2(b) is better than the one in figure 2(a), and both of them are much better than traditional delta-delta features.

Of course, it can be argued that the gain simply comes from the extra hidden layer or the increase of number of parameters, so we also experimented on original delta classified by a neural network with 3 and 4 hidden layers. From figure 4, we can see that the improvement of 3 layers from 2 layers and 4 layers from 3 layers are marginal. We believe this is due to the poor initialization of weights and the limited ability of SGD. However, we should notice that the new dynamic features are actually initialized and optimized in the same way. So we believe that the extra hidden layers are not the reason why the new delta features are better.

In all the experiments in this section, we tied the weights of transformation only within layer. Concretely, as shown in the right one in figure 2, the weights to get d_{t+1}, d_t and d_{t-1} from their corresponding static coefficients are forced to be the same, and can be different from the weights to calculate dd_t(from d_{t+1} and d_{t-1}). This is not mandatory in our framework. In some of our later experiments, we found out that no matter untying all the weights, tying weights within layer, or tying all the weights, the difference is negligible with new delta and new delta-delta features for recognition. So in all the following experiments, we still only tied our weights within layers for learning new dynamic features.

4.5. Phoneme recognition

In this section we tested the features learned in our framework in real phoneme recognition tasks. From the results in section 4.3 and section 4.4, we used fully connected matrices with linear activation function and an architecture like figure 2(b) to learn higher order of differences. Then the new features are used in a hybrid HMM-DNN system to get final recognition results.

In our framework, we are actually proposing a two-staged procedure (learn new differences and use them for recognition). We made it this way mainly because of the fact that the pre-training of DNN makes a difference in phoneme recognition task with TIMIT [14]. But if you have lots of data, we suggest training all the weights in one shot, since our frame can be easily extended to learning dynamic features for a context of several frames altogether. We also tried to learn new differences this way on TIMIT, and the result makes little difference with the above method that we learn new differences for each frame individually and then arrange them into contexts.

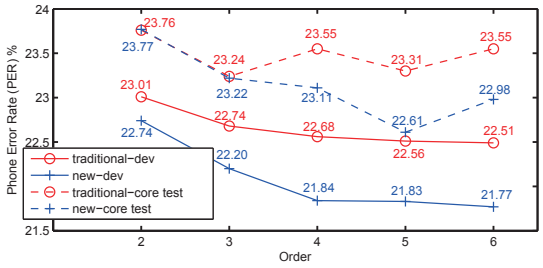


Fig. 5. Phoneme recognition results of different order of differences on development set and core test set of TIMIT by comparing traditional differences and new differences learned with our framework.

We compared the recognition results of the traditional way of calculating differences with our framework of learning differences from the 2nd order to the 6th order. All results are shown in figure 5. In each of our experiment, the number of order represents we use the basic 13-dimensional basic MFCC concatenated with the 1st order of difference up to the specified order, making them $(order+1)*13$-dimensional. We can see that with original differences, much higher order helps, but the improvement beyond 3rd order is quite marginal. Compared with the traditional way, the new differences learned with our framework consistently prevail. Also, we should notice that the gap between our method and traditional method increases with the order of difference used, which verifies the importance of variability our framework provided in capturing the dynamic information underlying the acoustic feature sequence.

5. CONCLUSIONS

In this work, we have presented a new framework with neural network to learn better dynamic features for phoneme recognition. We have shown in our experiments that the layers to obtain dynamic features should be full-connected with linear activation function. Furthermore, we have demonstrated that dynamic features learned with our framework are better than traditional dynamic features and such effect is even clearer when higher order dynamic features are used.

6. ACKNOWLEDGEMENTS

This work is supported by the National Basic Research Program of China (2012CB316401, 2013CB329304), the National Natural Science Foundation of China (61375027, 61370023), the Upgrading Plan Project of Shenzhen Key Laboratory (CXB201005250038A) and the Science and Technology R&D Funding of the Shenzhen Municipal.

7. REFERENCES

[1] H. Ney, "Acoustic-phonetic modeling using continuous mixture densities for the 991-word darpa speech recognition task," in *Proc. ICASSP*, 1990, vol. 90, pp. 713–716.

[2] J. G. Wilpon, C. H. Lee, and L. R. Rabiner, "Improvements in connected digit recognition using higher order spectral and energy features," in *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*. IEEE, 1991, pp. 349–352.

[3] C. H. Lee, E. Giachin, L. R. Rabiner, R. Pieraccini, and A. E. Rosenberg, "Improved acoustic modeling for continuous speech recognition," in *Proc. DARPA Speech and Natural Language Workshop*, 1990.

[4] R. Chengalvarayan and L. Deng, "Use of generalized dynamic feature parameters for speech recognition," *Speech and Audio Processing, IEEE Transactions on*, vol. 5, no. 3, pp. 232–242, 1997.

[5] A. Mohamed, G. E. Dahl, and G. E. Hinton, "Deep belief networks for phone recognition," in *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.

[6] G. Dahl, D. Yu, L. Deng, , and A. Acero, "Large vocabulary continuous speech recognition with context-dependent dbn-hmms," in *Proc. ICASSP*, 2011.

[7] G. E. Hinton, L. Deng, D. Yu, G.E Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainathand, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, 2012.

[8] L. Deng, J. Li, J. T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, et al., "Recent advances in deep learning for speech research at microsoft," *ICASSP 2013*, 2013.

[9] S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, "The htk book," *Cambridge University Engineering Department*, vol. 3, pp. 175, 2002.

[10] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using gaussian mixture speaker models," *Speech and Audio Processing, IEEE Transactions on*, vol. 3, no. 1, pp. 72–83, 1995.

[11] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for hmm-based speech synthesis," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*. IEEE, 2000, vol. 3, pp. 1315–1318.

[12] K. Richmond, "Preliminary inversion mapping results with a new ema corpus," 2009.

[13] I. Sutskever, J. Martens, G. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[14] A. Mohamed, G. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. on Audio, Speech and Language Processing*, 2011.

[15] K. F. Lee and H. W. Hon, "Speaker-independent phone recognition using hidden markov models," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 37, no. 11, pp. 1641–1648, 1989.