Adaptation of a feedforward artificial neural network using a linear transform

Jan Trmal, Jan Zelinka, and Luděk Müller
{jtrmal, zelinka, muller}@kky.zcu.cz

Department of Cybernetics, University of West Bohemia
306 14, Plzen, Czech Republic *

Abstract. In this paper we present a novel method for adaptation of a multi-layer perceptron neural network (MLP ANN). Nowadays, the adaptation of the ANN is usually done as an incremental retraining either of a subset or the complete set of the ANN parameters. However, since sometimes the amount of the adaptation data is quite small, there is a fundamental drawback of such approach – during retraining, the network parameters can be easily overfitted to the new data. There certainly are techniques that can help overcome this problem (early-stopping, cross-validation), however application of such techniques leads to more complex and possibly more data hungry training procedure.
The proposed method approaches the problem from a different perspective. We use the fact that in many cases we have an additional knowledge about the problem. Such additional knowledge can be used to limit the dimensionality of the adaptation problem.
We applied the proposed method on speaker adaptation of a phoneme recognizer based on TRAPS (Temporal Patterns) parameters. We exploited the fact that the employed TRAPS parameters are constructed using log-outputs of mel-filter bank and by virtue of reformulating the first layer weight matrix adaptation problem as a mel-filter bank output adaptation problem, we were able to significantly limit the number of free variables. Adaptation using the proposed method resulted in a substantial improvement of phoneme recognizer accuracy.

1 Introduction

Nowadays, the MLP ANN are increasingly used in the speech recognition field. Their uses include applications in speech recognition tasks, discriminative features production, language modeling, etc. The main characteristic is that the networks have very large number of parameters (hundreds of thousands or millions).

Given the size of the network, the training and retraining phases are computationally demanding and the amount of data needed during these phases is

* This research was supported by the Ministry of Education of the Czech Republic, project No. MŠMT LC536 and by the Grant Agency of the Czech Republic, project No. GAČR 102/08/0707 and by the University of West Bohemia, project No. SGS-2010-054. The access to the METACentrum computing facilities provided under the research intent MSM6383917201 is appreciated.

significant. This is not necessarily a problem during the training phase. However, there are situations, where the possibility of fast retraining on fresh data is beneficial. Moreover, the amount of the fresh data is usually quite small.

The limited volume of available data is an obstacle that renders the speaker adaptation and speaker adaptive training paradigms common in HMM ASR field very difficult to implement. The main issue is in the disbalance between the limited amount of available adaptation data (several hundred feature vectors) and large number of free variables. The common training algorithms under these conditions tend to heavily overtrain the network. This problem can be circumvented to some extend by employing cross-validation, early stopping and similar approaches, but it complicates the training process and increases the demands on amount of data.

In this paper we present a general method that enables adaptation of the weight matrix of the first layer of the ANN even if only a small amount of data is available. This is made possible by limiting the number of adaptation parameters. In many cases we can use an additional knowledge about the structure of feature vector, nature of the task, etc. to enforce an inner structure of the adaptation matrix and thus limit the number of free variables. Doing so enables substantial performance improvements through adaptation even on small data sets.

2 Multi-layer perceptron artificial neural network

Any forward operation of a L-layer MLP ANN can be described as follows

$$\boldsymbol{a_0}(k) = \boldsymbol{x}(k)\boldsymbol{W}_0 \tag{1}$$

$$\begin{aligned} \boldsymbol{y}_i(k) &= \boldsymbol{g}_i(\boldsymbol{a}_{i-1}(k)) \\ \boldsymbol{a}_i(k) &= \boldsymbol{y}_i(k)\boldsymbol{W}_i \end{aligned} \qquad i = 1, \ldots, L-1 \tag{2}$$

$$\boldsymbol{z}(k) = \boldsymbol{g}_L(\boldsymbol{a}_{L-1}(k)) \tag{3}$$

where the $D_i \times D_{i+1}$ matrices $\boldsymbol{W}_i$, $i = 0, \ldots, L-1$, are called weight matrices and the vector functions $\boldsymbol{g}_i$, $i = 1, \ldots, L-1$, are called transfer functions. The weight matrices are trained to minimize a loss function $\boldsymbol{E}$ that is usually of the following form

$$\boldsymbol{E}(\boldsymbol{Z}, \boldsymbol{T}) = \sum_{k=0}^{K} E(\boldsymbol{z}(k), \boldsymbol{t}(k)) \tag{4}$$

where K is the total number of training examples, the $K \times D_L$ matrix $\boldsymbol{Z}$ represents network outputs and the $K \times D_L$ matrix $\boldsymbol{T}$ represents the target values (teacher data). The pair of k-th rows of the matrices $\boldsymbol{Z}$ and $\boldsymbol{T}$ represents the k-th output $\boldsymbol{z}(k)$ and the target vector $\boldsymbol{t}(k)$.

The most usual choices of function E are E_{MSE} (i.e. mean square error) or cross-entropy E_{XENT}. See [1] for more info.

For the training there is a wide variety of methods to use. The most common one is the backpropagation and its modifications.

3 Adaptation of the neural network

The most straightforward approach is just to treat the adaptation data just like plain training data and the original set of weight matrices W_i, $i = 0, \ldots, Q$ as a "good initialization". Then adaptation of the old ANN means simply retraining the old ANN on new data.

We propose another approach, inspired by the weight-sharing approach used occasionally to improve generalization of ANN. Let's begin with expressing the new weight matrix $\boldsymbol{W}_0$ as

$$\boldsymbol{W}_0 = \boldsymbol{\Gamma}\boldsymbol{W}'_0 \tag{5}$$

where $\boldsymbol{W}'_0$ represents the old weight matrix and $\boldsymbol{\Gamma}$ is an *adaptation matrix*. The overtraining phenomena during the adaptation phase can be reduced or overcomed by virtue of enforcing an inner structure of the adaptation matrix $\boldsymbol{\Gamma}$, thus limiting the number of free variables that must be determined.

Suppose the D_0 by D_0 adaptation matrix $\boldsymbol{\Gamma}$. Because of its assumed inner structure, we can express the matrix as a function of a S-dimensional vector $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_S]$.

$$\boldsymbol{\Gamma} = \boldsymbol{\Gamma}(\boldsymbol{\gamma}) \tag{6}$$

or, expressed alternatively

$$\boldsymbol{\Gamma} = \begin{pmatrix} \Gamma_{11}(\boldsymbol{\gamma}) & \Gamma_{12}(\boldsymbol{\gamma}) & \ldots & \Gamma_{1D_0}(\boldsymbol{\gamma}) \\ \Gamma_{21}(\boldsymbol{\gamma}) & \Gamma_{22}(\boldsymbol{\gamma}) & \ldots & \Gamma_{2D_0}(\boldsymbol{\gamma}) \\ \vdots & \vdots & \ddots & \vdots \\ \Gamma_{D_0 1}(\boldsymbol{\gamma}) & \Gamma_{D_0 2}(\boldsymbol{\gamma}) & \ldots & \Gamma_{D_0 D_0}(\boldsymbol{\gamma}) \end{pmatrix} \tag{7}$$

Therefore, instead of computation of $D_0 \times D_0$ parameters we have to determine S parameters. Note that this is without any loss of generality, since S may be even bigger than $D_0 \times D_0$.

We want to minimize the criterion (4). Attempt to minimize it directly leads to application of either gradient-less optimization method (i.e. Powell's algorithm, Nelder-Mead simplex algorithm), or gradient method where gradient is replaced by its approximation. Both these approaches are usually a last-resort solution, because of their slow convergence and computation demands. Luckily, the expressions of gradient computations are quite easy to obtain.

Applying the matrix derivative chain rule for $\frac{\partial \boldsymbol{\Gamma}}{\partial \boldsymbol{\gamma}}$ we get

$$\frac{\partial E}{\partial \gamma_i} = \mathrm{Tr}\left[\left(\frac{\partial E}{\partial \boldsymbol{\Gamma}}\right)^{\mathrm{T}} \frac{\partial \boldsymbol{\Gamma}}{\partial \gamma_i}\right] \qquad \text{for} \quad i = 1, \ldots, S \tag{8}$$

where the computation of the expression $\frac{\partial \boldsymbol{\Gamma}}{\partial \gamma_i}$ is straightforward, since by definition the $\Gamma_{kl}(\boldsymbol{\gamma})$ is known for every element Γ_{kl} of the matrix $\boldsymbol{\Gamma}$.

The expression $\frac{\partial E}{\partial \boldsymbol{\Gamma}}$ can be determined by a similar approach as used for backpropagation. Using the equations (5) and (1) – (3) to compute $\frac{\partial E(k)}{\partial \boldsymbol{\Gamma}}$ and

applying the derivation chain rule, we arrive to the following expression

$$\frac{\partial E(k)}{\partial \boldsymbol{\Gamma}} = \frac{\partial E}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{a}_{L-1}} \prod_{i=L-1}^{1} \left(\boldsymbol{W}_i^{\mathrm{T}} \frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{a}_{i-1}} \right) \boldsymbol{W'}_0^{\mathrm{T}} \boldsymbol{x}(k) \tag{9}$$

The derivatives $\frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{a}_{i-1}}$ and $\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{a_{L-1}}}$ are $D_i \times D_i$ ($D_L \times D_L$ respectively) matrices $\frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{a}_{i-1}} = (\sigma_{kl})$, where the element σ_{kl} at coordinates (k, l) is given by

$$\sigma_{kl} = y_i \delta_{kl} - y_k y_l \tag{10}$$

in case when g_i is a softmax transfer function and

$$\sigma_{kl} = \delta_{kl} y_k (1 - y_l) \tag{11}$$

in case when g_i is a sigmoidal transfer function.

For the error function expression $\frac{\partial E}{\partial \boldsymbol{z}}$ the following expressions holds

$$\frac{\partial E_{\mathrm{XENT}}}{\partial z_{ij}} = -\delta_{ij} \frac{t_i}{z_j} \tag{12}$$

$$\frac{\partial E_{\mathrm{MSE}}}{\partial z_{ij}} = t_i - z_j \tag{13}$$

4 Phoneme recognition and traps parametrization

The used TRAPS feature vectors are constructed from the log-output of mel-filter bank. The process of the construction is described in detail in [2], assume here for the sake of simplicity the following approach. The process is depicted on figure (1).

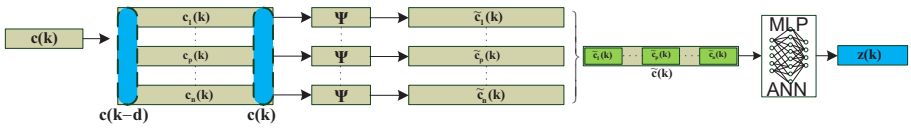

Fig. 1. A scheme of TRAPS phoneme recognizer

Assume for the given k-th frame of speech, the mean-normalized vector of log-outputs from the mel-filter bank is $\boldsymbol{c}(k) = [c_1(k), c_2(k), \ldots, c_N(k)]$, where N is the number of frequency bins.

The vector of D consecutive outputs of the p-th filter bank $\boldsymbol{c}_p(k) = [c_p(k - D + 1), \ldots, c_p(k)]$, $p = 1, \ldots, N$ is then decorrelated by a $D \times N_\Psi$, $N_\Psi \leq D$ matrix Ψ

$$\tilde{\boldsymbol{c}}_p(k) = \boldsymbol{c}_p(k) \cdot \Psi \qquad p = 1, \ldots, N \tag{14}$$

Usually, the Ψ matrix is a discrete cosine transform matrix. The vectors $\tilde{\boldsymbol{c}}_p(k), p = 1, \ldots, N$, are merged together, yielding the TRAPS vector $\tilde{\boldsymbol{c}}(k)$ of size M, $M = N_\Psi N$, $\tilde{\boldsymbol{c}}(k) = [\tilde{\boldsymbol{c}}_1(k), \ldots, \tilde{\boldsymbol{c}}_N(k)]$. The vector $\tilde{\boldsymbol{c}}(k)$ is then used as the input $\boldsymbol{x}(k)$ in expression 1. Therefore, obviously, $D_0 = M$.

The TRAPS feature vectors are usually quite long, since they span several hundreds of milliseconds of the original acoustic track. The features are fed into the ANN trained to produce phoneme posteriori probabilities. The ANN has two-layer design, with a sigmoid transfer function in the hidden layer and a softmax transfer function in the output layer.

5 Speaker adaptation of ANN

There is a significant variability among speakers. Among other reasons, the cause of variability is a different length of the vocal tract. The different length of the vocal tract manifests itself in shift of the formant center frequencies. There is a variety of methods that deal with this problem.

One of the simplest methods is called VTLN (Vocal Tract Length Normalization). VTLN applied on mel filters compensate the pitch shift by warping of the frequency spectrum. However, it has been shown (see [3] and [4]) that VTLN can be represented as a linear transform of the original, unnormalized coefficients. Therefore, the linear transform can be used for speaker normalization, even without linking it directly to VTLN. The normalized ("adapted") frequency bin $c_i(k)$ (using the notation from previous section) is obtained from the old ("unadapted") frequency bin $c'_i(k)$

$$c_i(k) = \sum_{j=1}^{N} \gamma_{ij} c'_j(k) \qquad i = 1, \ldots, N \tag{15}$$

and the coefficients γ_{ij} must be determined during the speaker normalization phase of the training process. As can be seen, the coefficients γ_{ij} form a $N \times N$ matrix $\boldsymbol{\gamma}$

$$\boldsymbol{\gamma} = \begin{pmatrix} \gamma_{11} & \gamma_{12} & \cdots & \gamma_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{N1} & \gamma_{N2} & \cdots & \gamma_{NN} \end{pmatrix} \tag{16}$$

Depending on the amount of adaptation data, we can limit the number of coefficients, starting from full matrix, going through various banded matrices and ending with a diagonal matrix.

From the description of simplified TRAPS construction process we can infer the structure of adaptation matrix $\boldsymbol{\Gamma} = \boldsymbol{\Gamma}(\boldsymbol{\gamma})$. The resulting matrix $\boldsymbol{\Gamma}$ will have a remarkably similar structure

$$\boldsymbol{\Gamma} = \begin{pmatrix} \boldsymbol{\gamma}_{11} & \boldsymbol{\gamma}_{12} & \cdots & \boldsymbol{\gamma}_{1N_\Psi} \\ \boldsymbol{\gamma}_{21} & \boldsymbol{\gamma}_{22} & \cdots & \boldsymbol{\gamma}_{2N_\Psi} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\gamma}_{N_\Psi 1} & \boldsymbol{\gamma}_{N_\Psi 2} & \cdots & \boldsymbol{\gamma}_{N_\Psi N_\Psi} \end{pmatrix} \tag{17}$$

however the matrix element $\boldsymbol{\gamma}_{ik}$ represents a $N \times N$ diagonal matrix

$$\boldsymbol{\gamma}_{ik} = \gamma_{ik}\mathbf{I} \tag{18}$$

where $\mathbf{I}$ is an $N \times N$ identity matrix.

Fig. 2. A scheme of possible ANN adaptation

The transformation of the vector $\boldsymbol{c}(k)$ by matrix $\boldsymbol{\gamma}$ is therefore equivalent to the transformation of the TRAPS vector $\tilde{\boldsymbol{c}}(k)$ by a matrix $\boldsymbol{\Gamma(\gamma)}$. Different approaches may be suitable in different cases (see fig. 2)

- Mel-filter bank fixed and ANN weights fixed – use matrix $\boldsymbol{\Gamma}$ to transform the TRAPS features $\tilde{\boldsymbol{c}}(k)$
- Mel-filter bank fixed and TRAPS fixed – use matrix $\boldsymbol{\Gamma}$ to transform $\boldsymbol{W}_0$
- TRAPS fixed and ANN weights fixed – use matrix $\boldsymbol{\gamma}$ to transform $\boldsymbol{c}(k)$

6 Experiments

The experiments were done on the telephone speech corpus SpeechDat-East. The corpus contains utterances from about 1000 speakers. For training of the ANN, we used only the phonetically balanced sentences (ID S0-S9, X0-1). Note that some speakers are not represented by a complete set of the 12 phonetically balanced sentences. The ANN topology was $330 \times 1500 \times 111$. The dimension of output layer reflects the phonetic alphabet consisted of 37 phonemes and each phoneme was modeled as a three state unit.

	$R = 0$	$R = 10$	$R = 100$	$R = 500$	$R = 1000$
band_all	75.30	75.30	75.54	75.92	75.87
band_nosil	**76.54**	76.45	76.30	76.11	75.63
diag_all	74.92	75.06	74.92	76.02	75.73
diag_nosil	75.87	76.07	75.92	75.49	75.49
diag_vocal	75.87	75.87	75.83	76.26	76.07

Table 1. Recognizer accuracy (Acc), evaluated including non-speech events, the baseline value is 75.63, R denotes different choices of the regularization constant

Since the speaker independent network was trained on the complete training part of the corpus, we decided to split the testing set. This was to ensure validity

of our experiments. From the testing data, we selected only 97 speakers that were represented by all 12 sentences. For each speaker, we divided randomly the appropriate set to a set of 10 utterances and a set of 2 utterances. The sets of 10 sentences were used as the adaptation data and the sets of 2 sentences as the test data.

The audio recordings are about 6–8 seconds long; however the utterances themselves are just about 2–5 seconds long. Taking only the actual speech into account, there was approximately 40 seconds of speech data available to adapt the neural network on. We experimented with different variants of adaptation matrix structure and with different choices of which phoneme classes the adaptation should be performed on.

7 Regularization

We studied the influence of regularization as well. To compute the regularization cost we used the following expression

$$E_r = |\boldsymbol{\gamma} - \boldsymbol{I}| \tag{19}$$

Since the optimal ratio between the training error and the regularization is not known, the E_r is combined with the expression (5) in the following way

$$\boldsymbol{E}(\boldsymbol{Z},\boldsymbol{T},\boldsymbol{\gamma}) = \sum_{k=0}^{K} E(\boldsymbol{z}(k),\boldsymbol{t}(k)) + R \cdot E_r(\boldsymbol{\gamma}) \tag{20}$$

where the constant R is called a *regularization constant* and is usually determined by choosing its value from a predefined set of weights.

7.1 Training process

Because we wanted to test the potential of the introduced approach, we used supervised adaptation of the network parameters. As the teacher data, we used the phone forced alignment of the reference transcript by an HMM alignment tool. After every update of weights, we performed another alignment run. We limited the number of update-realignment cycles to 4.

8 Results

The experiment results are shown in tables Table 2 and Table 1. The names of columns represent the individual combinations of the shape of the matrix $\boldsymbol{\gamma}$ ("diag" represents diagonal matrix, "band" represents tridiagonal matrix) and class of phonemes the adaptation has been performed on ("all" represent adaptation on all phoneme classes, "nosil" represents adaptation on all phonemes except silence and non-speech event classes, "vocal" represents adaptation only on voiced phonemes).

Since the neural network was done on the original TRAPS, we determined the matrix $\boldsymbol{\Gamma}(\boldsymbol{\gamma})$ first. The TRAPS are constructed from outputs of 15 mel-filters. This means that instead of adapting of $330 \times 1500 = 500k$ free parameters, we used only 15 free parameters with the diagonal setup, 43 parameters with the triagonal setup or 225 parameters for the full matrix $\boldsymbol{\gamma}$ setup. For

	$R = 0$	$R = 10$	$R = 100$	$R = 500$	$R = 1000$
band_all	73.20	73.11	73.32	73.66	73.45
band_nosil	**74.04**	73.78	73.87	73.49	73.32
diag_all	72.73	72.94	72.90	73.62	73.45
diag_nosil	73.32	73.49	73.24	72.99	72.94
diag_vocal	73.41	73.28	73.49	73.83	73.57

Table 2. Recognizer accuracy (Acc), evaluated excluding non-speech events, the baseline value is 73.07, R denotes different choices of the regularization constant

the best combination of regularization constant and adaptation setup, we performed the Wilcoxon signed rank test under the null hypothesis that median of the difference between the unadapted and adapted networks is zero. The p-value was $p = 0.003$, which is sufficient to reject the null hypothesis at the level $\alpha = 0.003$.

9 Conclusion

In this paper we devised a novel approach to a MLP ANN adaptation. We applied the presented method on speaker-based adaptation of a phoneme recognizer based on MLP ANN. Adaptation of this kind of networks on small amount of data is generally a difficult task because of quite large number of network parameters. Application of the method lead to a significant reduction of the number of free variables, thus alleviating the overtraining problem. On approximately 40 seconds of speaker data we achieved absolute improvement of approximately 1% (4% relative reduction of phone error rate).

References

1. Bishop, C.M.: Neural networks for pattern recognition. Oxford University Press, ISBN 0-19-853864-2 (2005)
2. Schwarz, P., Matějka, P., Černocký, J.: Towards lower error rates in phoneme recognition. Lecture Notes in Computer Science **2004**(3206) (2004)
3. Pitz, M., Molau, S., Schlüter, R., Ney, H.: Vocal tract normalization equals linear transformation in cepstral space. In: Proceedings of EuroSpeech 2001. (2001) 2653–2656
4. Pitz, M., Ney, H.: Vocal tract normalization as linear transformation of MFCC. In: Proceeding of EuroSpeech 2003. (2003)